Technical Report
CMU/SEI-92-TR-6
ESC-TR-92-006

Carnegie-Mellon University
Software Engineering Institute

Proceedings of the CASE Management Workshop

Cliff Huff
Dennis Smith
Ed Morris
Paul Zarrella

September 1992

ADA258234

Technical Report
CMU/SEI-92-TR-6
ESC-TR-92-006
September 1992

Proceedings of the CASE Management Workshop

Cliff Huff
Dennis Smith
Ed Morris
Paul Zarrella

CASE Technology Project

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Table of Contents

List of Figures

List of Tables

Proceedings of the CASE Management Workshop

Abstract: The Software Engineering Institute (SEI) Computer-Aided Software Engineering (CASE) Technology Project sponsored a workshop to address a number of key CASE management issues. The workshop was held at the SEI in Pittsburgh, Pennsylvania on June 19-20, 1991. At the workshop, a representative group of SEI affiliates from industry, government, and academia discussed among themselves such management topics as CASE acquisition policy, what CASE tools can and cannot do, CASE and metrics, and CASE tool selection. The results of these discussions are summarized in this report.

1 Introduction

There are a wide range of issues that management must deal with when addressing the incorporation of new computer-aided software engineering (CASE) technology into their organization. Some of these key issues were the topic of discussion at this CASE Management Workshop held at the Software Engineering Institute (SEI) on June 19-20, 1991. The specific areas for discussion at this workshop were:

- CASE Acquisition Policy
- What CASE Tools Actually Do – What They Don't Do
- CASE and Metrics
- CASE Readiness
- CASE Tool Selection

This workshop was the second in a series of CASE-related workshops sponsored by the CASE Technology Project at the SEI. This workshop gathered 45 professionals from industry, government, and academia with a common interest in CASE and CASE management.

2 Keynote Address

In the introductory keynote address, Dr. Bill Curtis, Director of the Software Process Program at the SEI, spoke on "*Where's the Leverage for Improving Software Development–An Empiricist Talks CASE.*" This address and the commentary that followed were based on his considerable experience in examining the software development process. His primary theme was that software productivity, quality, and costs cannot be explained outside the context where software engineering is performed. Software engineering technology (e.g., CASE) only has benefit through its impact on actual behavior during software development.

The outline of his talk is presented below:

- Review of traditional CASE acquisition process
- Description of a process-based approach to incorporating CASE
- Examination of software productivity
- Examination of what software designers do
- Discussion of "Is team design an oxymoron?"
- Review of the process focus on software development
- Discussion of process-based technology supporting process maturity levels 2 and 3 activities
- Discussion of process and CASE futures

3 Executive Overview of CASE Management Workshop

This second CASE Workshop sponsored by the SEI yielded a number of insights and guidelines to aid SEI affiliates in their efforts to integrate CASE technology effectively into their organizations. In some cases the sessions had implications for additional work and future research. One such topic is the CASE Production Efficiency Index developed and discussed by the metrics session. Summary results from each of five CASE Adoption Workshop sessions are presented below.

3.1 CASE Acquisition Policy

Due to ongoing and new government initiatives in the areas of CASE and environment technology, this session centered around the problems of establishing appropriate government policies for tools and environments. Topics covered by the discussions included:

- What are the current problems with government acquisition and policies for CASE tools?
- Can a uniform toolset be identified?
- What type of government CASE policy is appropriate?
- How can we encourage contractor and government cooperation?
- What are contractor and government responsibilities?

3.2 What CASE Tools Actually Do—What They Don't Do

This workshop session was devoted to creating a realistic assessment of the current capabilities of CASE tools. Participants discussed what CASE can do and cannot do, both in the near term (less than or equal to five years) and far term (ten years). Specific areas of discussion of what CASE actually does included:

- Enforcement of product standards
- Automation of the software process
- Re-engineering, reverse engineering, and restructuring support
- Tool interoperability
- Automatic code generation
- Data collection and communications

3.3 CASE and Metrics

This session was tasked to design an association of metrics with the use of CASE. The most significant result of this workshop group is the identification of an appropriate "toolkit" of design schema for addressing the problem of CASE and metrics. This toolkit draws upon the fields of economics and operations research, and provides a theoretical basis in the form of a *production efficiency index* (PEI) for the interpretation of data gathered through metrics.

3.4 CASE Readiness

Barriers to CASE are not limited to technology issues. Just as critical, if not more so, are issues related to the organization's readiness to adopt CASE tools and technology as well as the process maturity of the organization. The theme of the CASE Readiness session was measuring organizational readiness to adopt CASE tools. This workshop session aimed to identify:

- Major factors that influence organizational CASE readiness
- Approaches for obtaining insights into where a particular organization fit with respect to factors identified above
- Formulation of consensus responses to the process-specific questions such as "What is the impact of the maturity level on the usage of specific tools?"

3.5 CASE Tool Selection

The goal of this workshop session was to examine CASE tool selections issues and to provide some practical advise on CASE tool selection criteria and methodology. Tool selection from a high-level process abstraction is basically compose of the following elements:

- Process and methodologies
- Strategy
- Selection of individual tools
- Adoption of tools

To narrow the discussion, this group reached a consensus to focus their efforts on developing a set of tool selection strategies. These strategies would be aimed at a high level of strategic tool selection criteria. These strategies are topics to consider in selecting tools and could become portions of an organization's selection process, as appropriate. Considerations at three levels of organizational hierarchy were discussed: project, organizational, and enterprise.

4 CASE Acquisition Policy

4.1 Introduction

The CASE Acquisition Policy session of the CASE Management Workshop began with Joe Morin of the SEI providing a presentation of potential topics in CASE acquisition and policy. Potential topics outlined in Joe's presentation included:

- Setting realistic expectations for CASE acquisition
- Getting the right methods and tools
- Adopting CDRLs to tool capabilities
- Comparing the buyer's and vendor's perspectives on CASE acquisition

The open discussion following Joe's presentation addressed many of these issues, as well as a wide range of additional issues. In light of ongoing and new government initiatives in the areas of CASE and environment technology, discussion centered around the problems of establishing appropriate government policies for tools and environments. Questions covered by the discussions included:

- What are the current problems with government acquisition and policies for CASE tools?
- Can a uniform toolset be identified?
- What type of government CASE policy is appropriate?
- How can we encourage contractor and government cooperation?
- What are contractor and government responsibilities?

4.2 Problems With Government Acquisition and Policies

Currently, government contracts are awarded based on the technical and economic merits of the proposal with little regard to the tool support to be used by the vendor in building the system. Winning contractors for the many government projects use widely diverse software methods and tools. Unfortunately, the quality and productivity of the tool support used by these contractors varies.

The variety and quality of methods and tools used by various vendors has profound affects on the maintenance of software after it is delivered to the contracting agency. Not only must the government support these many systems, but in many cases it must attempt to do so using the tool suite selected by the original contractor. Where the tools and methods are inappropriate, inadequate, or nonexistent, a tremendous burden is assumed by the maintaining agency. This scenario is problematic for a number of reasons:

- The initial choice and burden of tool acquisition is on the contractor. The government can only hope that the contractor makes decisions that are appropriate for long-term maintenance of the system.

- Selection of appropriate tools that assist in good quality development and maintenance of software requires knowledge of the process and methods of both the contractor and the government. Unfortunately, since tool needs are so closely tied to an organization's process and methods, it is unlikely that the tools chosen by the contractor will fit with the needs of the maintaining agency. It is also clear that many contractors do not understand the post deployment software support (PDSS) needs of the government.
- When the government has attempted to specify tools or types of tools to be used during product development, the contractor will often pay "lip service" to the request and make token use of the tools. The group noted that in fairness to the contractor, it was stated that there is a tremendous risk involved in adopting a new tool. The adoption process is expensive and time consuming, and it does not guarantee success. In many cases, the most prudent decision from the development (but perhaps not the maintenance) standpoint is to rely on existing tools and methods.
- There is little experience on the part of the government or contractors to translate the extra costs and risks of enforcing the use of new tools into actual dollar figures. It has not been completely determined who should pay for the adoption costs and accept potential risks.

4.3 Identifying Uniform Toolsets

One approach that has been suggested is to select a set of approved tools, which are then used by all contractors for development, and subsequently by the government for system maintenance. The topic of tool selection was discussed at the workshop session, and a number of important considerations were identified. These considerations include:

- **The cost of the tool relative to the impact (cost/benefits).** Tool types that were identified as having a high cost/benefits ratio include configuration management tools, change management tools (those that can perform requirements traceability and impact analysis), and program generators (tools that can be used to create other tools).
- **The application domain.** It was felt that the domain of the software system plays a major role in influencing the type of tools that are applicable. Only a relatively small subset of tools (such as documentation support tools) spans application domains.
- **The tool platform.** Although a greater number of tools are migrating toward "open systems" platforms, there is by no means a universal platform in either the commercial or DoD world. Diverse platforms include a variety of personal computers, workstations, and mainframe computers.
- **The special needs and processes of an organization.** Few (if any) tools can support the needs of a wide variety of organizations. For example, even within the DoD, different agencies have different processes, documentation requirements, and security requirements. No one tool can address all of these needs.

- **The universality of the tool.** In spite of the difficulty in identifying a "universal" tool, the potential for sharing of a tool across multiple projects must be considered. It is impractical for every new maintenance software component to use unique tools due to associated licensing and training costs. An "optimal" tool must meet the needs of a significant subset of projects.
- **The market viability of the tool.** The tool market is new and dynamic. The collapse of tool vendors is frequent. Unfortunately for DoD systems, where software is often maintained for long periods, the collapse of a vendor during the long maintenance period can be a catastrophe.
- **Tool procurement, operation, and maintenance costs.** It is becoming increasingly difficult to provide for the upkeep of many tools. Licensing schemes that provide for flexibility in usage patterns are essential.
- **The quality of the tool support available.** As tools increase in complexity, they require more training and stronger customer assistance. Often the quality of this assistance can differentiate between an unsuccessful and successful tool adoption.
- **The individual features of the tool.** The quality of a tool's features are important, particularly in influencing users to adopt the tool. It is important to realize, however, that tool features represent only one element in a larger list of important tool characteristics.
- **The strength of the tool vendor's commitment to emerging capabilities and standards .** Examples of such standards and technologies are PCTE, the ECMA/NIST reference model, object oriented technology, and reuse technology. While it is unlikely that a tool will meet or provide all of these standards/ technologies, a measure of vendor interest may be participation at relevant meetings.

In light of the many considerations, and the diverse range of systems and organizations involved in DoD development and maintenance, it was determined that no single set of tools could meet the needs of all tool users. Attempts to mandate tool usage have been largely unsuccessful. Common experience suggests that users will not necessarily use a tool, even if it is made available (or even mandated). Some interesting approaches have been used by organizations wishing to provide tools at low costs to government users (such as the NASEE toolset), but it is clear to the organizers of these efforts that other factors such as tool adoption play a major (and perhaps dominant) role in success.

4.4 Appropriate Government CASE Policies

In contrast to the general belief that mandating of tools does not work, workshop participants felt that a carefully conceived policy that encouraged tool usage and identified standards where appropriate could be useful in furthering contractor and DoD tool usage. The characteristics of this policy include:

- The facilitation of a higher level of standardization, such as uniform framework service support similar to that identified in the ECMA/NIST reference model.
- A approach to encouraging commercial investment in tools and tool standards.
- The development of a short-term and long-term vision of tool usage by government contractors and government agencies.
- The identification of open architecture standards to be supported by the various government services. These standards would be developed cooperatively with the commercial world.
- A set of broad guidelines for government agencies on the procurement and insertion of tool and environment technology.
- A set of procurement guidelines that assist government agencies in specifying tool and environment platforms, appropriate levels of process support, and mechanisms for evaluating the tool support in proposals. This action plan should be certain to address:
  - Ways of identifying "tool tokenism" in proposals. Tool tokenism refers to the common practice of including mention of tool usage in proposals with no firm commitment to the tools.
  - Ways of determining the underlying process support provided by a contractor's proposed toolset.
  - Guidelines for required demonstrations of tool capabilities as a factor in contract award.
  - Guidelines for the development of tool usage scenarios and sample problems as factors in contract award.
  - Guidelines for determining whether environment, tool, and process support identified in proposals is appropriate for the "receiving" (often maintenance) organization. Note that the receiving organization is often different than the contracting organization.
- An action plan identifying how government personnel will be trained to use new procurement guidelines to evaluate proposals. This action plan should also identify how tool and environment expertise could be developed within the government, or contracted externally.
- A method for evaluating a potential contractor's tool capabilities, perhaps in relation to the organization's process capabilities as measured by the SEI Capability Maturity Model.

4.5 Encouraging Contractor and Government Cooperation

A critical factor in any DoD plan to improve tool support is to encourage improved tool support among government contractors. Without improved support, upgrading of government capabilities will have reduced impact due to the commonly recognized difficulty of maintaining software with a toolset different than that used in software production.

A government plan to encourage commercial investment might promote contractor CASE tool acquisition. It was suggested that a major reason for poor tool penetration in the government was poor tool penetration among contractors producing government software. A number of suggestions were provided to help encourage government contractors to increase their investment in tool technology, including:

- Assist senior executives of government contractors in recognizing the need for a greater capitol investment in software production and maintenance capabilities. Software engineering continues to be capitalized far below the level common in low technology professions. In addition, senior executives must recognize that many of the barriers to increased capitol investment are internal.
- Leverage the existing SEI assessment process to encourage corporations to invest in tool support. The carrot and stick approach embedded within the SEI process assessment capability has been instrumental in generating awareness of the importance of software process.
- Develop a tools database which helps both government and the commercial sector in evaluating and procuring tools in a timely fashion.
- Encourage the use of CASE technology by offering incentives to contractors that successfully use tool technology. One possibility is modifying the contractor rating process to include use of tool technology. A second possibility involves developing tax incentives or rebates for investment in tool technology. Still another approach might involve government funding for the procurement and adoption of selected tools within industry.
- Gather data to demonstrate to commercial organizations that there is a long-term payoff in using CASE tool technology. This data does not exist (for the most part) in the government sector, but is potentially available in the commercial world. While it may be difficult to encourage government contractors to relinquish data which could help competitors, the SEI is ideally placed to gather and report this data in a manner that does not violate an organization's confidentiality.
- Modify the format and contents of RFPs to better encourage creative and effective tool solutions.
- Generate sophisticated plans for technology insertion in government and industry. Almost universally, technology transition is identified as a major cost (and barrier) to tool adoption. A transition organization such as the SEI may be well positioned to assist.
- Develop mechanisms that allow contractors to work with the government in evaluating, selecting, and transitioning tool support. For example, government and IBM technical people have worked with a vendor technical group to gain exposure to available tool technology.
- Motivate organizations to excel in tool and process automation by developing awards to reward exceptional performers (something like the Malcolm Baldridge award).

- Initiate an ESPIRIT like cooperative project involving the government and cooperations and task it with identifying better process, tool, and environment support.

4.6 Government and Contractor Responsibilities

A final question addressed in the acquisition workshop session concerned who should assume primary responsibility for working the variety of tool issues confronting the government and contractors. The recommendations of the workshop session participants follow from the basic and widely held belief that while more advanced processes and methods are employed in industry, it is important for government organizations to influence the type of tool support used in order to insure applicability to government software development and maintenance needs. Thus, the resulting list reflects both where the expertise resides, and what the government must do to insure access to best appropriate methods and tools.

- The primary decision on process and methods is best left in the hands of industry. Since the adoption of a new method or tool can introduce considerable risk into a development effort, the government may be best served by the use of familiar methods and tools. In addition, it is unlikely that any process or method mandated by the government will be embraced by industry.
- Industry should prioritize its needs in terms of new process and methods. This will assist the government in encouraging research and development of new methods and tools that meet the needs of both government and industry.
- The government, because the its unique, high leverage position, should make significant efforts to justify the need for better tooling and encourage the use of such tools.
- Industry must evolve toward the use of better software processes. New and improved processes will have a significant impact on the future evolution of tools and environments.
- Industry must resolve the remaining platform standardization issues. Government should play a role as a member of standards groups in order to insure that government needs are met. However, it should not take the leading role.
- Industry should continue work to converge on appropriate mechanisms for tool integration. The limits of the current integration work (such as PCTE, ATIS, and SoftBench) should be investigated.
- Mechanisms to provide tool education, training, and support should be developed in industry.
- The government should invest resources in demonstrations of new concepts which might be underfunded in industry. In effect, the government should mitigate some of the risk of tool adoption in industry by providing examples and data supporting tool use.

- The government should encourage the development of methods and tools that simplify the transition from software development to software maintenance.

5 What CASE Tools Actually Do—What They Don't Do

5.1 Introduction

This workshop session was devoted to creating a realistic assessment of the current capabilities of CASE tools. Participants discussed what CASE can and cannot do, both in the short-term (five or fewer years) and long-term (ten years). Section 5.2 will discuss what CASE tools can do (both now and in the future). Section 5.3 will identify things CASE cannot do. Section 5.4 provides a timeline identifying the current and future capabilities of CASE.

5.2 What CASE Tools Actually Do

5.2.1 Enforcement of Product Standards

Current CASE tools are able to enforce a limited range of product standards. The product standard capabilities of CASE tools are developing rapidly, but remain relatively inflexible. Among major product standards supported by CASE tools are:

- Most CASE analysis and design tools can help ensure adherence to a particular (tool supported) method. Unfortunately, the methods supported by tools are not easily tailorable to a particular organization. The tools are relatively inflexible both in the sequencing of activities allowed, and in the actual standards enforced.
- Many CASE tool vendors claim that their tools are capable of generating documents compliant with relevant standards (like MIL-STD-2167A). In reality, such claims are too broad. The tools provide only limited, semi-automated support for generation of documents. In many cases, what is provided is a template containing relevant tool data, and stubs for information which is customarily maintained in different formats (such as documentation systems or project management tools).
- CASE tools are also available to audit source code and identify tool compliant and non-compliant code segments. The standards supported by code analysis tools are often based on well known heuristics for code quality. The tools usually offer some degree of flexibility in interpreting heuristic data, but support for unique standards required by a particular organization remains outside the capability of most tools.

In the short-term, it is expected that CASE tools will support increasing degrees of user customization of both methods and standards. Also, as links between various tools become better established, it is expected that document production will become increasingly automated, with the eventual goal of fully automated document generation. In the more-distant future, it is expected that CASE tools will support multimedia capabilities and standards.

5.2.2 Automation of the Software Process

Currently, CASE tool support for the software process is extremely limited. CASE tools are able to support only a limited number of methods, and can provide automated recording and auditing of a few activities. The reasons for limited process support are twofold:

- Immaturity of the software engineering discipline, which leads to a lack of acceptable software process models to automate
- Immaturity of CASE tools and software environment frameworks which support tool integration and a software process.

In the near future, it is expected that methodological support offered by CASE tools may be tailored to the user's needs. Automated auditing and recording will encompass a wider range of activities and life cycle phases, tools will generate useful reports based on audit trails.

Support for notification between modules (essential for software process support) is now becoming available in environment frameworks. As process support mechanisms develop in environment frameworks, it is hoped that simultaneous work will lead to the identification of well understood and accepted process models. It is expected that in the near future, operational process environments will become available, and conformance to the standard process can be assured by CASE tools and environment frameworks. In the more distant future, the software engineering process will be automated in a way that supports the manner in which software engineers work.

A fear was expressed, however, that automated process support might lead to unnecessary and potentially damaging process restrictions, particularly if the process required is a strict waterfall model. It is hoped that in the more distant future, process support will allow development to proceed in the way in which developers actually work, that is, bouncing between waterfall stages. It is also hoped that ultimately process support will be offered for more specialized, application-specific development and maintenance models.

5.2.3 Re-engineering, Reverse Engineering, and Restructuring Support

At present, CASE tools support some problems in reverse engineering, re-engineering, and restructuring, but in a limited way. From input code artifacts, tools can generate useful functional and structural views of a system. Unfortunately, there is little support for the generation of data models from code. Code restructuring capabilities can be useful, but are better developed for management information systems than for other sorts of systems.

During the next five years, it is expected that tools will be developed, which allow existing applications to be manipulated at the design level. This will occur by reverse engineering source code to generate design information, directly modifying the design information, and then regenerating the application.

It is also hoped that over this period, tools will begin to develop mechanisms for the recovery of the design decisions made. Such capability may develop via the integration of reverse engineering tools with tools manipulating design documents and project histories. Unfortunately, tools will never be able to recover information that is not appropriately recorded. Since so many of the critical data and decisions of a project reside only in the minds of the engineers, reverse engineering tools may never provide a satisfactory rendering of project history.

In the more distant future, it is hoped that development and maintenance of programs will occur at a higher level of abstraction, where the engineer will seldom deal directly with source code. In this scenario, the design of the software can be used to regenerate the functional specifications, as well as to generate the actual source code. Ultimately, a representation of the functional specifications can be directly manipulated by engineers. From these specifications, a design and source code can be automatically created.

5.2.4 Tool Interoperability

Currently, tools are at best partially interconnectable. This interconnectivity relies on the hooks encoded into individual tools, and therefore it varies considerably. What connections do exist tend to be unidirectional. For example, systems exist where modification of a design will automatically cause regeneration of source code (most often in the form of specifications). However, few (if any) systems exist where modification of the source code leads to corresponding changes in the design.

The reasons for the current limitations on tool interoperability are many, but include:

- The proliferation of interconnectivity standards. There are currently many competing standards that offer some degree of tool interconnectivity. Some of these standards are pushed by individual vendors, while others have support of (constantly changing) industry groups. It is difficult, if not impossible, for a tool vendor to support all such standards.
- The poor interoperability between different tool vendors. In light of the many competing standards, this is not surprising. What is interesting, however, is that competing vendors are using the "poor" interoperability of the competitor's tool as a device to sway perspective customers. The customer is left to determine which of the competing claims is most valid. The reality, unfortunately, is that even tools that claim they are interoperable allow only the most rudimentary forms of integration.
- The poor interoperability between life cycle phases. Tools (and methodologies) that are specifically geared to a life cycle phase often provide few automated (or even theoretical) links to tools supporting different life-cycle phases. As a result, engineers must mechanically "shoehorn" the output of one tool into the input stream of a tool supporting the subsequent life cycle phase. This activity often requires extreme effort to develop a logical link, along with massive reentering of data.

- The proprietary nature of tool architectures. Only a small portion of tool vendors are willing to fully disclose how their tool works, and how to access portions of a tool. Without full disclosure, it is very difficult for a third party to integrate two or more tools from different vendors. While vendors are providing more and better programmatic interfaces, it remains difficult to dissect a tool to a level where it can be effectively integrated.

In the short-term, it is expected that a usable repository and object base for software engineering data will be developed. Even so, in the near future, many CASE tools will continue to use their own databases. Additionally, control integration facilities allowing tools to send and receive messages are likely to be developed. These advances will converge with a number of standards efforts and will lead to the development of usable environment frameworks. Such frameworks will provide a focal point for tool developer, and lead to increasing interoperability of tools. As tools are ported to the emerging frameworks, bidirectional links will be established between tools.

In the more distant future, the emerging frameworks will become accepted as standards. A large cadre of tools will be interoperable, and bidirectional links between tools will become better established. In summary, an integrated software engineering environment will become a reality. With a well established framework, tool vendors can develop a tool suite utilizing the presentation, control, and repository services of a framework.

5.2.5 Automatic Code Generation

Currently, automatic code generators exist for MIS-type applications, and code "stub" generators exist for a variety of applications. Where code generation capabilities do exist for DoD applications, they are often immature for the following reasons:

- DoD applications are more difficult to automate for a variety of reasons including the complexity and uniqueness of hardware interfaces, the severe resource constraints (particularly timing constraints), and the state of the art nature of many applications.
- An immature understanding of the methods and techniques necessary to adequately capture and express design. Without an adequate vocabulary for expressing design, it is not possible to capture all of the information necessary for complete and accurate code generation.
- Lack of control of code optimization. Code produced by code generators often cannot meet the tight constraints imposed by systems.

Code reuse techniques, now in their infancy, may one day automate the process of generating applications from design or possibly from specifications. Unfortunately, code reuse is not currently widely accepted or supported, in part due to a lack of cultural acceptance of reuse.

Soon, it can be expected that mechanisms for synchronizing design information and code will be enforced. This synchronization will include the automated evaluation of design quality and expressibility in code. Synchronization will also extend to the automated control of system configuration and generation of appropriate make files.

In the more distant future, it is expected that system development and maintenance will be performed at an increasingly abstract level. Debugging and automatic optimization will occur based on requirements based constraints. Source code will be generated or assembled from components using design level reuse.

5.2.6 Data Collection and Communication

Only a small percentage of the engineering artifacts of a software project are currently saved in tool databases. What is stored is distributed across multiple tools. The picture that can be drawn from available data is at best inconclusive, and at worst misleading. This lack of a complete picture of the software engineering process has led in part to a number of problems, including the inability to determine project state and identify project risk. It has also contributed to the common fear among software engineers concerning the misuse of performance data.

It is expected that in the short term, CASE technology can contribute to the generation and maintenance of a more accurate and complete set of engineering artifacts. It is expected that a repository will act as a "living" document from which a more accurate view of project status can be determined. Since the development of appropriate process models appears to be lagging slightly behind the development of repository technology, it is likely that views of the data at this point will be largely ad hoc. In the more distant future, as increasingly complex process models are developed, the views offered of the data in repositories will be formalized into different roles.

5.3 What Tools Will Never Do

As significant as the role of tools may eventually be, there are a large number of individuals and activities which cannot be replaced by tools. Unfortunately, some organizations invest in tools as a method of overcoming deficiencies in a wide variety of areas. It was clear to workshop participants that CASE tools could not perform the following functions:

- Minimize or simplify the intellectual rigor and insight needed to specify, design, implement, or maintain complex, quality software.
- Fix organizational problems, or overcome a poorly construed or developed process.
- Measure or ensure the overall quality of the process or product, as determined by the users
- Do the difficult parts of reverse or re-engineering that require insight into an engineer's motivations for a specific software architecture.
- Provide platform, tool, and development phase interoperability without additional support from a framework or environment

It is important when evaluating vendor claims that an organization understand both the current and future capabilities and limitations of CASE tools, in order that an informed decision be made.

5.4 CASE Capability Timeline

The following two tables provide a summary of the session's judgement on the timeline of current and future capabilities of CASE. This timeline is divided into 5-year increments from 1992 to 2001. Estimates in the following CASE areas are given:

- CASE support for *Re-Engineering, Reverse Engineering, and Restructuring*
- CASE enabling and formalizing *Software Process and Methods*
- CASE enabling *Process*
- CASE *Interoperability* with respect to *Platforms, Tools, and Life-Cycle Phases*
- CASE enabling *Product Standards*
- CASE support for *Automatic Code Generation*
- CASE facilitating *Communications and Data Collection*

In the following two timeline tables, plain text bulleted items denote CASE capabilities, while italicized items denotes observations or concerns.

	Reverse Engineering	Process and Methods	Enable Process	Interoperability Platforms, Tools, and Phases
1992				
	• Reverse engineering in functional and structural views • No support for reverse data-modeling • Code restructuring for MIS is more advanced	• Enforce some methods (e.g., structural analysis) • Automatic auditing when product is checked in (POC) • Automatic recording when activity is completed (POC)	• Automate notification between modules with FRAMEWORKS	• Partial connectivity (Hooks) • Unidirectional with respect to phase
	• *Cost and Complexity of "META Tools"*	• *Only some methods supported* • *Lack of tool interoperability*	• *Immaturity of Software Engineering Discipline*	• *Proliferation of standards* • *Poor interoperability between different vendors* • *Poor interoperability between life-cycle phases* • *Proprietary architectures*
1996				
	• Design level manipulation of existing code • Recovery of design decisions (what and why) • Good data reverse engineering tools	• User specified tailoring of methods • Automatic auditing of products (real proj) • Automatic recording when activity is completed (real proj) • Useful reports from completion notices	• Operational process environments • Development of specialized processes • Conforming to standard processes	• Repository • Persistent Object Base Convergence of Standards • Usable Frameworks • Usable Bidirectional Links
		• *Proliferation of methods*	• *Fear of process restrictions*	
2001				
	• Programming and maintenance at higher level of abstraction (source code transparent) • Abstraction of design to functional specification	• Parts of process automated • Support for the way an engineer really works	• Development will be allowed to proceed in the way that people actually work (bouncing between waterfall phases) • Specialized application processes	• Hidden Frameworks • Acceptance of Standards • Total interoperability of platforms • Bi-Directional

Table 5-1: CASE Timeline Part A

	Product Standards	Automatic Code Generation	Communication Data Collection
1992	• Capability to audit code (coding standards) • Adherence to methods (balancing, etc.) • Semi-Automatic Document Production	• Generators exist today for MIS type applications • Code "stub" generators exist for a variety applications	• Partial engineering artifacts in database
	• *Locked into implementation of Method*	• *Reuse not widely accepted nor supported* • *Crude – lack of optimization need to meet tight constraints* • *Immature reuse technology – lack of cultural acceptance* • *Immature understanding of how to capture and express design*	• *Fear of misuse of user performance data*
1996	• User-specified tailoring of Method • Automatic document production to standards	• Synchronization of design and code enforced • Include automatic evaluation of design quality • Include automatic generation of make files • Well integrated with configuration management	• Real/Complete engineering artifacts (Repository acts as living document, different views of data are ad-hoc)
2001	• Evolution to alternative media standards	• Development and maintenance are done at an abstracted level (code hidden, abstract level debugging, automatic optimization based on requirements-based constraints) • Design level reuse	• Automatic views of data generated for different roles

Table 5-2: CASE Timeline Part B

6 CASE and Metrics

6.1 Introduction

In his keynote address, Dr. Curtis described the process of design as occurring within three conceptual spaces: the problem domain space, the design schema space, and the solution space. The design process matches domain expertise and a problem statement (from the problem domain space) with a toolkit of design schemas and design expertise (from the design schema space) to express a problem solution (the solution space). Events which occurred on the first day of the CASE and Metrics working group session reinforce this model of the design process.

The CASE and Metrics group was tasked to design an association of metrics with the use of CASE. The topic of CASE and metrics has received considerable attention at similar workshops because CASE adoption requires significant investments which quantitative analysis can help to justify. Yet while participants in these workshops may have had significant problem domain expertise (i.e., CASE and software engineering expertise), they may not have had access to the appropriate "toolkit" of design schemas.

The most significant result of this workshop group is the identification of an appropriate "toolkit" of design schema for addressing the problem of CASE and metrics. This toolkit draws upon the fields of economics and operations research, and provides a theoretical basis in the form of a *production efficiency index* (PEI) for the interpretation of data gathered through metrics. Measures of PEI apply generically to any kind of production processes, but need to be calibrated for particular kinds of processes. The calibration of the PEI model to construct a *software production efficiency index* (SPEI) requires software engineering expertise.

A calibrated SPEI is necessary to evaluate the impact of CASE on production efficiency, since it provides the vehicle for separating the impact of CASE from other factors, such as product complexity, experience of personnel, development environment characteristics, etc. Additionally, the SPEI can be used to:

- Evaluate the impact of CASE on fine-grained production processes, such as design processes, coding processes, and testing processes.
- Evaluate the impact of other factors on production processes, such as training, process improvements, hardware improvements, etc.
- Identify the key factors that have an impact on production efficiencies as a basis for computing the return on investment (ROI) for production improvements.
- Provide a national measure of software productivity.
- Provide a vehicle for sustained refinement of SPEI measurements over time, analogous to the way the consumer price index (CPI) and gross national product (GNP) measures evolve over time.

Section 6.2 of this summary report describes how the workshop arrived at the SPEI approach. Section 6.3 describes what the SPEI is, how it is measured and how it is calibrated. Section 6.4 discusses the limitations of SPEI. Finally, Section 6.5 concludes with recommended next steps.

6.2 Interacting Dimensions

The initial focus of discussions was the FURPS (for "Functionality, Usability, Reliability, Productivity, Stability) model [FURPS 90]. FURPS defines a one dimensional partition of metrics that can be combined with orthogonal partitioning schemes, for example, phases of a software life cycle.

Classification schemes (such as FURPS versus Life Cycle) provide one means of reducing the complexity introduced by the many and varied kinds of metrics data that can be collected. Two IEEE standards, *Standard for Software Productivity Metrics* (IEEE-P1045/D4.0) and the draft *Standard for a Software Quality Metrics Methodology* (IEEE-P1061/D21), are reflective of the richness in kinds of metrics available for collection. Yet it is precisely this richness which led to a group consensus that discussions about what kinds of metrics to collect would be unproductive, or at least duplicative, of activities such as the IEEE metrics standards.

Several members of the group also expressed skepticism regarding the validity of two-dimensional classification schemes such as FURPS versus life cycle; indeed, it was generally agreed that there were several dimensions, some of them interacting. One important dimension addresses the motivations for gathering metrics. For example, an organization attempting to use metrics to support an improvement in process maturity from level 1 to level 2 [Humphrey 89] might adopt a different set of metrics and metrics gathering techniques than an organization attempting to evaluate the ROI of investing in a specific class of CASE tools (for example, testing tools).

The discussions of metrics rationale led to the hypothesis that it should be possible to discuss metrics in terms of an analogue to the requirements/design/implementation paradigm of systems development. Requirements corresponds to rationale, that is, *why* metrics will be gathered, design corresponds to choosing *which* metrics will support an objective, and implementation corresponds to deciding *how* the metrics will be gathered. It was hoped that separating why, what, and how from each other would provide some insight into a problem area that is well understood in one way (what metrics can be collected), and little understood in another way (which measures are valid, what does the data mean?).

The group brainstormed about rationale for gathering metrics, and arrived at a list of thirty reasons. Later analysis revealed that three broad classes of reasons were discussed. One class was process-focused, especially with respect to process control and visibility. The second class was product-focused, especially with respect to the "ilities," that is, reusability, maintainability, reliability, etc. The third was metrics-focused, and addressed the concern that a body of data needs to be collected to support the empirical analysis of which metrics are meaningful, and how they are related to each other, and to the underlying software production process. Table 6-1 summarizes the results.

Process Oriented	Product Oriented	Metrics Oriented
Check consistency of process	Adhere to standards	Provide data baseline
Evaluate effect of change	Evaluate product usability	Perform experiments
Evaluate points of impact	Identify need for maintenance	ID, quantify factors
Evaluate refinements	Support analysis (regression)	ID correlations
Measure response to change	Estimate need for re-work	
Estimate time and effort	Estimate value/impact of reuse	
Identify, predict bottlenecks	Software release decisions	
Quantify progress to goals	Determine potential for reuse	
Risk assessment		
Predict costs and schedules		
Evaluate what's right, wrong		
Justify investments		
Estimate production functions		
Provide feedback for different audiences		
Provide feedback and communication		
Provide insights for improvements		
Troubleshoot		
Compare group performances (teams, agencies		

Table 6-1: Three Classes of Metrics Rational

Unfortunately, the partitioning approach shown in Table 5-1 did not provide the hoped-for concrete vehicle for identifying which metrics should be used under which circumstances. It did, however, spark interesting discussions that raised a number of questions, including:

- What measures are related to ROI? Isn't ROI relative to time frames and objective measures of return? Do well-accepted objective measures exist? For example, is SLOC a valid, useful objective measure?
- How can you be sure what you are measuring? For example, can you separate domain knowledge from use of CASE tools and demonstrate which had greater impact?
- Are all measures by nature indirect? That is, do you measure CASE effectiveness by measuring product qualities? by measuring productivity? How are these related?
- Are concepts such as "productivity" sufficiently well-defined, or are such concepts composites of many measures, that is, source lines of code (SLOC), number of tasks completed per unit time, number of errors per 1000 SLOC produced, etc., which are not uniformly well defined?
- How do various measures interact? For example, will higher productivity in terms of SLOC per month have detrimental impact on product "ilities?"

During these discussions, a theme that frequently re-emerged was the way various measures interacted; also important were questions about whether it would be possible correlate measures to each other (and to some desired effect, such as productivity increases).

It was at this point in the group discussions that the concept of production efficiency measures was introduced by Mr. Suresh Konda. This concept emerged as the central theme for the remaining discussions.

6.3 The Software Production Efficiency Index (SPEI)

Production efficiency indexes (PPI) are a means of measuring complex production processes that depend upon many input and output factors. A classical example of a PPI is the consumer price index. The PPI provides a mathematically sound vehicle for expressing the relationships among well-defined input and output factors, and has applicability to the measurement of any production process.

In retrospect, the group's early discussions were based upon a skewed view of metrics as being applicable only to output measures, that is, SLOC, number of errors detected during regression testing, dollar-cost per SLOC, and so forth. Instead, the PEI, represented schematically in Figure 6-1, includes the input factors which are the basis for interpreting the meaning of the output measures.

One point Mr. Konda made was that although, ideally, input factors are orthogonal (that is, non-correlatable) to each other, as are output factors, through use of statistical devices input factors and output factors may be correlated. For example, it should be possible to correlate dollars invested in CASE to a product measurement such as reliability.

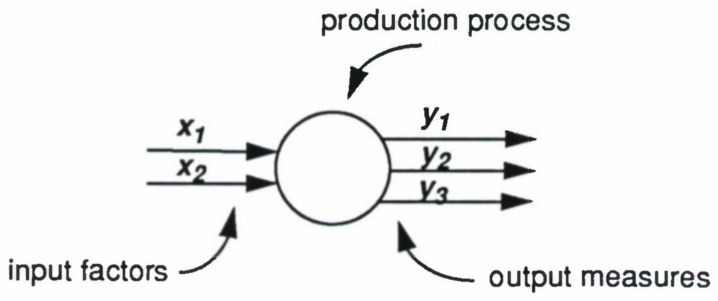


Figure 6-1: Production Processes

What makes measurement of software production efficiency difficult is that input measures are in fact not likely to be orthogonal to each other. For example, an individual's expertise in a problem domain may have multiple effects when combined with training in the use of a particular design methodology suited for the problem domain. The introduction of non-linearity means that a generic PPI model will not suffice; instead, a specialized PPI is needed, as is expertise from software practitioners to identify, and help quantify, interactions among input factors.

One specialized PPI, called the Software PEI (SPEI) in our session, is proposed and is represented algebraically in Figure 6-2.

$$\frac{g(\underset{\sim}{y})}{f(\underset{\sim}{x})} \equiv \frac{\sum_{j \in y} c_j y_j}{\sum_{i \in x} a_i x_i + \sum_{k \in x} b_{ik} x_i x_k}$$

Figure 6-2: Software Production Efficiency

Several important points of note about the equation in Figure 6-2 are:

- The coefficients of input factors *x* and output factors *y* serve two purposes: to assign weights to various factors, and to convert measures into dimensionless scalars.
- To support calibration of the model the need to correlate one dimension of input factors (e.g., dollars invested in CASE) to changes in the SPEI between two SPEI observations (SPEI *deltas*).
- The denominator shows non-linear interactions occurring between pairs of input factors, but such interactions could as well occur between triples, 4-tuples, and so forth.
- Figure 6-1 is not necessarily the most appropriate one for SPEI; the important point is the expression of non-linear interactions among input dimensions.

Examples of input factors include: budget constraints, deadline constraints, requirements (that is, use of particular standards), software development environment, etc. Examples of output measures include: SLOC, number of errors per 1000 SLOC, cost per SLOC, percent of code reused from other sources, number of Pepsi's consumed by developers, and so forth.

One very interesting result of the workshop was the realization that the SPEI could be applied to fine-grained processes as well as to the life cycle as a whole. For example, SPEI can be applied to individual steps in the life cycle. Figure 6-3 illustrates this possibility.

Several key points are illustrated in Figure 6-3:

- Application of SPEI measures to fine-grained processes simplifies the analysis of input/output correlations by reducing the size of input and output factors.

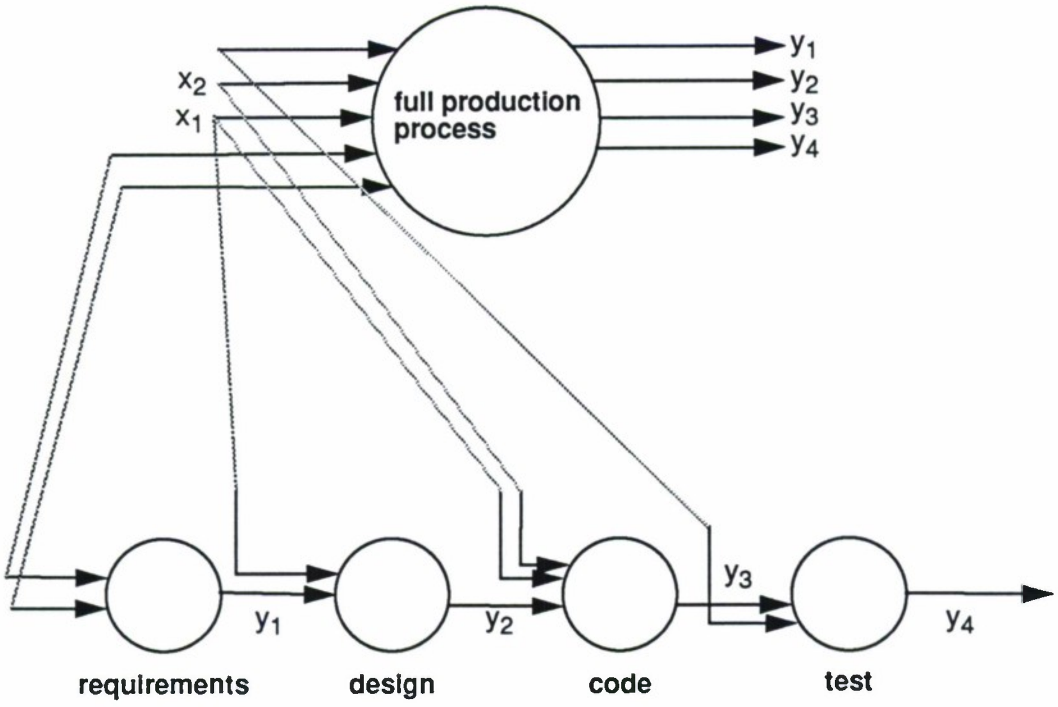


Figure 6-3: Application of SPEI to Fine-Grained Processes

- The reduced number of input factors also means that control over extraneous input factors is simplified, making possible more precise measures of impact due to individual factors. For example, products by different vendors could be compared within the limits of a fine-grained process.
- Input factors may apply to more than one fine-grained process (e.g., factor x_1) or may apply to only one fine-grained process (e.g., factor x_2).
- The output factors of one process can be thought of as input to the next process, that is, number of requirements as output of the requirements process and input to the design process.

6.4 SPEI Limitations and Caveats

Although the SPEI has many commendable features, it does not provide a universal framework for the analysis and use of metrics. Most significantly, the SPEI approach does not address issues of process dynamics. That is, the SPEI is a *post hoc* measurement of a process that has run to completion; it is not useful for process troubleshooting, nor does it make visible running processes or provide control over them.

Another limitation of the SPEI, which should also be considered as a caveat in the use of SPEI and perhaps other metrics models, is that it does not measure individual performance, but instead measures group performance. Thus, an output measure such as errors per 1000 SLOC can be deduced at the level of individuals, but this output measure would have as much to do with distant input factors as with individual performance. For example, a "botched" design could result in excessive failure rates detected at testing time.

Yet another caveat in the application of metrics-gathering regimes is that the metrics program must be uncoupled from incentives, at least initially. Failure to do so will almost certainly lead to skewed results as data is "fudged" and development processes become tailored to generating output results optimized for specific measures (for example, complexity measures).

6.5 Conclusions and Next Steps

This method of indexing efficiency during software production provides a theoretically sound basis for understanding and analyzing the interactions among various input factors of a production process, and correlating input factors to output factors. The applicability of SPEI to fine-grained production processes means the method can be applied surgically, and can be used to evaluate more precisely the impact of changes to input factors.

The SPEI does not address issues of process dynamics. In particular, it can not be used on in-progress processes for the purposes of troubleshooting, except to the extent that this is possible through application of SPEI to fine-grained processes within the context of a spiral life-cycle model. The SPEI is also not effective at measurement of individuals; rather, it is a measure of group performance (e.g., team, agency).

Several steps must be taken to apply the SPEI approach:

1. Software engineering expertise must be applied to create a baseline model of input factors, and their interactions. The IEEE metrics standards IEEE P1045 and IEEE P1061 are valuable starting points for this activity.
2. Output measures for quantitative and qualitative evaluation of software engineering products and processes must be agreed upon. The IEEE metrics standards, and the SEI Software Process Program's Process Metrics Project are valuable starting points for this activity.
3. A data "baseline" must be established. At this point it is important to collect data, even from an imperfect model of input/output factors and correlations. Imperfections in the model can be addressed by modifying the model at later stages; however, these modifications must preserve the correlation of observations across time and space (i.e., across sampling sites).

7 CASE Readiness

7.1 Introduction

The theme of the CASE Readiness session was measuring organizational readiness to adopt CASE tools. The theme revolved around a number of questions, including:

- Is there a relationship between CASE success and process maturity?
- Are certain tools more useful at different levels of process maturity?
- Does CASE adoption help to define the process?
- How, in terms of dimensions and metrics, is organizational readiness measured?
- What is the impact of the maturity level on the usage of specified tools?

7.1.1 Goals

The goals of this workshop session included:

- Identify major influences on an organization's readiness for CASE.
- Identify approaches which determine the readiness of a particular organization according to those influences.
- Formulate consensus responses to the questions introduced in Section 7.1.

7.1.2 Process

After participants introduced themselves and explained their areas of interest, the facilitator, David Kitson, explained the session themes and goals. In the course of discussion, workshop participants formulated both desired and realistic outcomes. To ensure meaningful results, all agreed to focus chiefly on major factors which influence organizational readiness; the remaining goals listed above were to be discussed as time allowed.

There was also a consensus that we needed to agree on a definition of CASE for the purposes of the workshop session discussion. Some time was spent reviewing the various reference documents provided the CASE Management Workshop binder. Initial discussions of the articles occurred, followed by a brainstorming session to determine organizational readiness factors. The objective was that all participants should play an active and equal role in the discussion. Therefore, a round robin approach was utilized for the initial brainstorming session. The results of the brainstorming session were reviewed and then categorized into an existing CASE Implementation Methodology with some modification. This also resulted in top level parameters for organizational readiness.

7.2 Discussion and Results

There was little doubt that CASE tools and technology can have a significantly positive impact on an organization's product, cost, process, methods, and environments, but it was agreed that in most organizations, such impacts have yet to be realized and that it will take much more time. Often, organizations have failed to use their CASE tools fully, and so have incurred great expense. Frequently these costs and failures have been due to inflated vendor claims, unreasonable user expectations, and the lack of the organizations readiness.

There are many factors that contribute to successful adoption of CASE tools and technology. The group noted that there are a number of unresolved problems in the areas of tool technology and integration which over time must be solved before an overall goal of integrated CASE tools and environments can be reached. It was beyond the scope of this session to address the other issues related to CASE success.

The following two additional documents were provided by participants, also authors, for review:

> Aharonian, L. K., *Preventing Expensive CASE Tool Shelfware*, IEEE CASE '90 Workshop, 1990.
>
> Yeh, R. Y.; Naumann, D. A.; Mittermeir, R. T.; Schlemmer, R. A.; Sumrall, G. E.; LeBaron, J.T., *COSMOS: A Commonsense Management Model for Systems*, IEEE Software, November 1991.

The CASE definition agreed upon for the Readiness session was:

> Any computer software application that assists development, management, and support personnel in the software development life cycle.

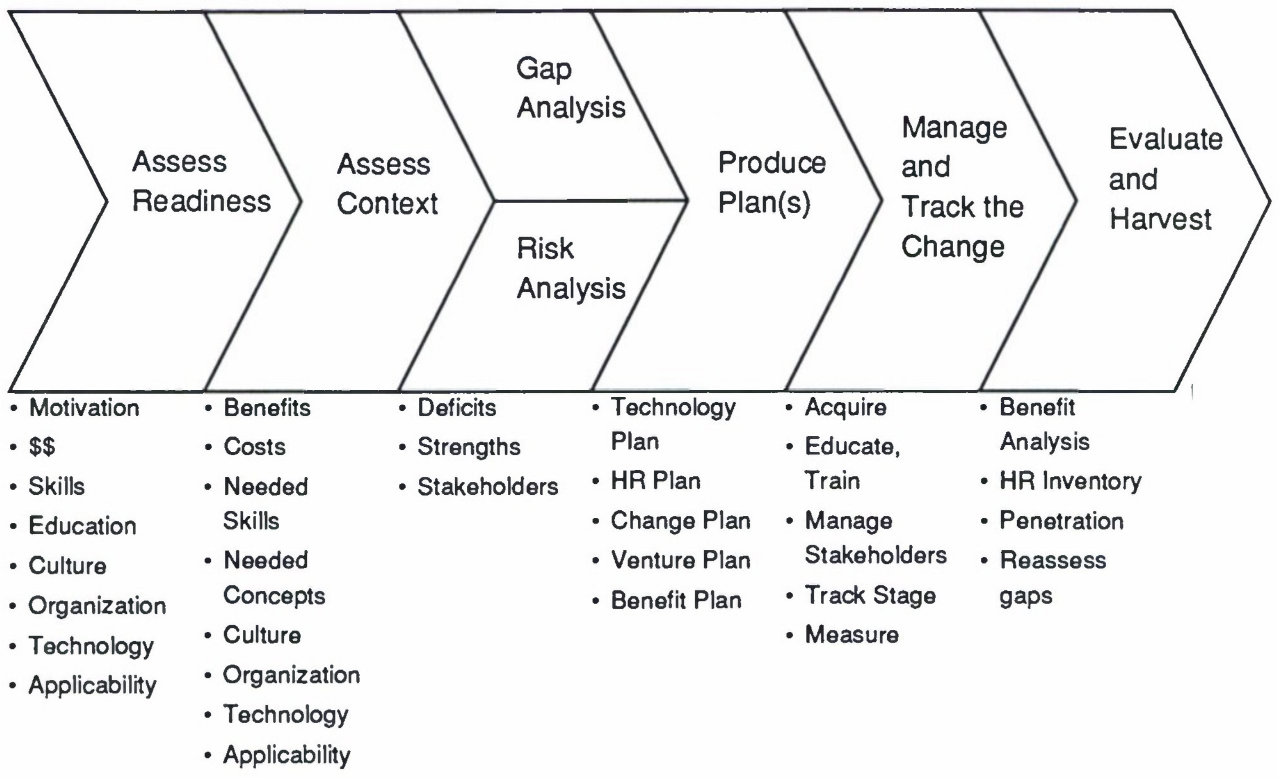
After the initial brainstorming session, we began to use as our baseline document, the article entitled "How to Become a Software Engineering Big Foot" by Howard A. Rubin. We reviewed and discussed the CASE Implementation Methodology put forth in two articles by Rubin [Rubin 90] [Rubin 91] and the issues raised by Dan Mosley [Mosley 89]. Rubin claims that an organization's readiness is a key to understanding CASE implementation, and he explains a multi-dimensional model for describing implementation methodology.

The readiness attributes identified in the brainstorming session were streamlined to eliminate duplication and overlap. We then attempted to see if all factors could be categorized by the dimension attributes defined by Rubin [Rubin 90] [Rubin 91] in the readiness footprint. We found that by augmenting several attributes and adding process, our results fit the Rubin model. The nine dimension attributes identified were from Rubin, except for the attribute Process which we felt was important to the organizations readiness. The nine dimension attributes are:

1. Motivation
2. Investment

3. Skills
4. Education
5. Culture
6. Organization
7. Technology
8. Applicability
9. Process

To provide more insight into the session, we have reproduced the CASE Implementation Methodology from Rubin [Rubin 90].



Source: [Rubin 90]

Figure 7-1: CASE Implementation Methodology

The following definitions, taken from Rubin [Rubin 90] [Rubin 91], clarify the first phase of the methodology outlined above.

Motivation	Intensity of drive to improve quality and productivity
Investment	Willingness to spend money to make software engineering happen
Skills	Ability to use conceptual foundations as a basis for performing work
Education	Knowledge of abstract conceptual platforms for contemporary skills
Culture	Risk aversion
Organization	Mechanism for technology transfer and support
Technology	Technology infrastructure in place today
Applicability	Work focus (new development or maintenance)

The group concluded that an organization's process was a key attribute in its readiness to successfully adopt CASE. An organization has to have a process that is ready to introduce and implement CASE, otherwise it may be best not to waste time and money on implementing tools, technology, and organizational change as it may have little chance of success. Discussion on whether an organization had to be at a certain maturity level, as defined by the SEI occurred. No consensus was reached. It was a general feeling that much depended upon the type and sophistication of a tool as well as the maturity level of an organization. If properly planned and implemented, CASE tools can benefit organizations at different maturity levels. Benefits will increase as an organization moves up the maturity model and as tools become more sophisticated and integrated. CASE tools that automate common activities are more quickly assimilated into an organization than CASE analysis and design tools, which require an even greater organizational readiness.

7.3 Top-Level Organizational Readiness Parameters

From our brainstorming list also emerged top-level organization parameters for readiness, which are identified below:

Activity

- Process and methodology
- Product and resources
- Technology

Communications	• Complexity of methodology
	• Head Count
	• Formal communications
Infrastructure	• Cultural communications
	• Individual/organization/project
	• Management and education

7.4 Dimension Attributes

The categorizing of readiness factors within the dimension attributes resulted in the following:

7.4.1 Motivation/Commitment

- To what extent that management is committed to making improvement?
- Does management believe status quo is sufficient?
- Are there competitive pressures that force adoption of CASE tools?
- Is there a need for improved communications and accuracy with CASE tools providing a competitive edge
- To what extent do top management recognize CASE tools as a strategic source of competitive advantage?
- What is the need for "quality" management of the product?
- Is there appreciation of realistic expectations from all levels of management?
- Is there long-term commitment from all levels of management?

7.4.2 Investment

- Required dollar investment/per person?
- Hardware vs. software investment required?
- Top management commitment of total investment (i.e., dollars, people, project)?
- Cost-benefit return period?
- Past return on investment results?
- Inventory of existing tools and hardware?

7.4.3 Skills

- Inventory of current skill levels
- Inventory of current abilities
- Current personnel levels of assignments
- Ability to learn new skills (are they teachable?)

- Organizational support for learning new skills (cross training)

7.4.4 Education

- Current level of conceptual knowledge
- Manager and peer attitude towards education
- Organizational support (intellectual and financial) for educational programs and training
- Educational programs (in-house training)

7.4.5 Culture

- Willingness to innovate and change.
- Ability to manage innovation and change.
- Tone of CEO (what kind of example is set?).
- How were failures handled?
- Recent organizational experience with innovation attempts, successes, failures, and rewards.
- An organizational structure to help manage innovation attempts.
- Willingness to incur risk.

7.4.6 Organization

- Infrastructure support (library, employee training, technical support [e.g., process group, technology transfer])
- Communications structure (degrees of interaction inside and outside the organization)
- Cohesion (organization has a shared vision)
- Reporting structure (hierarchical vs. network structure)
- Policy/procedure - Review policy and interview procedures

7.4.7 Technology

- Platform scale (mainframe, workstation, PC's) will affect tool selection
- What network capabilities exist (distributed, internal, external)
- Vendor profile, mix and match, what technology are they familiar with, are you prepared to integrate tool if required
- What kind of support tools are available and what is their potential integration
- Maturity of the present technology
- Operating version compatibility - may be using the right technology but not the right flavor

7.4.8 Applicability

- What is the normal product/domain for this organization?

- Extent to which similar products are produced by the organization (degree of commonality)
- What phases of the process is the organization responsible for
- What size projects by resource loading and duration per phase are normal
- What size projects are delivered by SLOC
- Who is the normal customer
- Is development normally co-located
- How knowledgeable, experienced and decisive is the software management
- How complex is the process by phase
- Methodologies employed
- Does a physical DB exist for documentation and /or code (potential reuse)

7.4.9 Process

- Location on the maturity model
- Process type (e.g., waterfall, spiral)
- Process drivers (to meet MIL SPEC 2167A might restrict tool select)

7.5 Conclusion

Barriers to CASE are not limited to technology issues. Just as critical, if not more so, are issues related to the organization's readiness to adopt CASE tools and technology as well as the process maturity of the organization.

8 CASE Tool Selection

8.1 Introduction

In his keynote address, Dr. Curtis gave a thumbnail overview of the tradition CASE Acquisition model. The model was depicted as:

- Experience a software failure
- Read ads in *Datamation* and *Computerworld*
- Purchase CASE tools from vendors
- Figure out what processes fit the CASE tools selected
- Assure each other the problems are over
- Blame staff for not using tools properly
- Hire consultant
- Become disillusioned

While this model does not fit all cases, it is more likely than not to be true. One can see in this model that CASE tool selection appears to be an ad hoc process. It is also a process which is subject to a great deal of variability during a successful CASE adoption experience. The goal of this workshop session was to examine CASE Tool Selections issues and to provide some practical advise on CASE Tool Selection criteria and methodology.

8.2 Initial Background Discussion

Prior to specific discussion, workshop attendees received a general overview of some important topics in the CASE selection process. The following sections highlight these topics and detail related issues.

8.2.1 The CASE Selection Process

Organizations should consider the following issues when determining which needs make the purchase of CASE tools necessary.

- What problem are you solving?
- What types of tools are available?
- Are there any “Showstopper” issues?
- Is there a need to narrow the focus of the selection process?
- What is the need for hands-on evaluations?
- What are the important decision and adoption considerations?

By focusing on these issue, an organization can efficiently direct its efforts to rapidly identifying potentially useful CASE tool candidates.

8.2.2 Identifying the Problem

You should define your problem as exactly as possible before buying a CASE tool; otherwise, you risk wasting money and leaving your organization's actual problems unsolved. However, if the norm in the software development world holds, it is likely "that 2 years after acquisition 70% of the tools are no longer used, and for those still in use, only 10% of the intended audience is using them in the proper manner" [Rubin 91]. In the long run, we hope with the robust tool section and adoption process, an organization will do significantly better than the apparent norm.

the problem which you identify may depend on the following influences:

- **Model of software development.** This refers various life cycle models such as a waterfall or spiral model of software development. Which model do you follow? Does a potential candidate CASE tool support that model of development?
- **Required tasks.** What specific tasks are you attempting to streamline and automate with the adoption of a CASE tool?
- **Leverage in tool support**. Does this CASE tool provide the amount of leverage you require? If not, you should examine tools of more or less sophistication, as appropriate.
- **Balance of Costs and Benefits.** To determine whether a CASE tool is worth the investment, you should arrange for a cost-benefit analysis. Your organization's professionals in corporate finance can advise and assist you. Involving them from the beginning can make your analysis more accurate and your ultimate choice of a CASE tool better informed.
- **Potential Risk.** You should assess how incorporating a new CASE tool may affect the cost, schedule, and performance of a project's required tasks.

8.2.3 Identifying the Types of Tools That Are Available

The CASE market place is quite diverse with a large and ever growing list of products (see Appendix A. CASE Market Overview for more details). There is a reasonable amount of summary material on existing CASE products, available from both independent commercial and government sources. Some of it varies in quality and detail, but as compared to gathering this data on their own, most organizations should find this material more cost effective.

The CASE Technology Project maintains a list of CASE Resource pointers. These pointers offer many different sources of information on CASE tools. This resource, while not all inclusive, does represent a significant cross section of the types of information available from commercial and government sectors. Workshop attendees received a current version of this list.

8.2.4 Sample "Showstopper" Issues

Showstopper issues are those issues that, by themselves, can cause a CASE adoption effort to fail. Listed below are some potential issues which can derail the successful use of many CASE tools.

- **Integration.** A CASE tool may be very difficult or impossible to integrate with other critical tools in your environment.
- **Scalability.** The CASE tool may perform satisfactorily on a small amount of code, but when used with the anticipated code of significantly greater size, the tool may perform unacceptably slow.
- **Cooperative processing.** A CASE tool may be a single user only tool and operate poorly, if at all, in a multi-user environment.
- **Process support**. A number of CASE tools incorporate some form of software development process, while others are process-independent. Depending on the needs of organization, it might be essential for a tool to follow critical standards and processes. If the CASE tool is not customizable, this may pose a serious problem.
- **Vendor stability.** Quite often, an exceptional CASE tool may be coupled with a vendor who may have a questionable long-term future. Most tools, however, have a long lifetime within an organization, so it is important to choose vendors who support their tools over the long term.

8.2.5 Hands-On Evaluations

Many organizational evaluation become mired in too much detail during the earlier stages of tool evaluation. We feel it is important to identify a very small number of evaluation criteria which will act as a high-level filter for selecting tools prior to in-depth evaluations. A previous SEI technical report, *A Guide to the Classification and Assessment of Software Engineering Tools* (CMU/SEI-87-TR-10), discusses some potential criteria. These high-level criteria include:

- Ease of Use
- Power
- Robustness
- Functionality
- Ease of Insertion
- Quality of Support

For those interested in commercially available electronic databases of tools and their characteristics, we currently know of two sources:

- **CASE OUTLOOK Guide to Products and Services (1991).** This publication includes a PC-DOS program called TOOLFINDER which allows users to selection from 20 major categories of CASE tool attributes. Overall the TOOLFINDER catalogs some 440 possible CASE tool details for over 850 CASE related tools. TOOLFINDER is designed precisely for locating CASE products that meet a handful of key but broad criteria.
- **CASEBASE by P-Cubed Corporation.** CASEBASE is a detailed electronic catalog on approximately 250 CASE products. CASEBASE contains an extensive repository of information on each product. CASEBASE permits comparison of products in 7 major categories and according to 182 features. Additionally, CASEBASE provides access to vendor-provided, product-related news releases, information on CASE-related articles, books, and other published materials plus a calendar of CASE-related events such as conferences, expositions and symposia.

Once the evaluation process begins, you will need to determine which criteria best suit your needs. iA good example of detailed evaluation criteria for a single class of CASE tools is embodied in a tool report entitled *Requirements Analysis & Design,* prepared by the Software Technology Support Center in 1991.

8.2.6 Decision and Adoption Considerations

Of the many decision and adoption considerations which CASE implementation raises, you should keep in mind some of the most important:

- **Staffing and training**. What sort of background does your staff have in the methodologies embodied in the CASE tool under consideration? Are the users proficient in using the methodology and user-interface which the CASE tools employs? The gap between the current skill level and the required skill level will need to be filled by an appropriate degree of training.
- **Piloting**. Do you want to test the selected CASE technology and implementation process in the form of a pilot project? If so, the selected pilot project should be representative of the other projects that are likely to use the new CASE technology.
- **Evaluating.** Regardless of how you implement a CASE tool, whether using a pilot project or directly introducing it into the organization, you should develop a set of success criteria before you begin. These criteria will help you to evaluate the overall success of your efforts.

8.3 Selected Focus Area

Based on a high-level process abstraction, tool selection is essentially composed of the following steps:

- **Process and Methodologies**. Identify or select and then document your organization's software development processes and methodologies.
- **Strategy.** Determine an overall strategy for automating those processes and methodologies.
- **Selection of Individual Tools.** Select individual tool that support your process and methodologies.
- **Adoption of Tools.** Purchase and install the tool(s) of choice, train the organization, manage the organizational changes brought about by the tool introduction, and analyze the effectiveness of tool usage with an eye towards fine tuning the tool and/or the organization for increase effectiveness.

After a period of initial discussion, the group reached a consensus to focus their efforts on developing a set of tool selection strategies. These strategies would be aim at a high level of strategic tool selection criteria. These strategies are topics to considered in selecting tools and could become portions of an organization's selection process, as appropriate. Considerations at three levels of organizational hierarchy were discussed. These levels were project, organizational and enterprise.

(While this session could have easily focused upon detailed selection criteria for CASE tools instead, but this was believed to be an unnecessary and inappropriate. This was due partly from the fact that several organizations like the STSC have already developed detail selection criteria for some different classes of CASE tools. Therefore work of this nature could prove to be largely duplicative.)

8.3.1 Definitions

For the purposes of this workshop session, the following definitions of project, organizational and enterprise were used:

- **Project**. A team dedicated to unified task or job. A task-directed entity with cost, schedule and performance responsibilities. Projects are the end-users of tools.
- **Organization.** An entity within a corporate enterprise, for example, a division or department with responsibilities across more than one project.
- **Enterprise.** A company or corporation or a DOD level organization. They have responsibilities across more than one organization.

8.3.2 Assumptions

To set the stage for the group discussions, we assumed the following:

- An organization has already a set of specified processes and methods for which they want to provide automated support.
- An organization may or may not have an existing Software Engineering Environment (SEE).
- There are already lists of issues for individual tool selection.

- The previous CASE adoption workshop, and current CASE readiness group already have addressed general tool adoption issues.
- Key issues have been identified, but the process itself has not yet been defined.

8.4 Strategy Focus Area Discussion

In the Strategy area of tool selection, we focused upon the following elements:

- Problem space determination
- Acquisition strategy
- Adoption strategy
- Readiness
- Process Control and Enforcement
- Software Engineering Environment (SEE)
- Standards
- Standard Practices

In subsequent paragraphs, we detail each of these elements. First, we provide definitions as appropriate. Second, we highlight important sub-issues in those elements. Third, we relate each strategy element to three levels of an organizational hierarchy: project, organizational, and enterprise.

8.4.1 Determine Problem Space

Definition Determining the problem space is aimed at identifying those specific tasks that a potential CASE tool may improve. Therefore, CASE tools should be bought primarily to solve specific problems.

Sub-Issues Here are some important reasons to seek out a CASE tool:

- Provide a completely automated or partially automated solution to a specific task.
- Insure adherence to standards, process and methods.
- Provide an enabling technology to improve quality.

(Note: readers are encouraged to examine Section 5, "What CASE Tools Actually Do—What They Don't Do," as a helpful source for additional insights on where CASE usage is appropriate.)

Scope

Project

- They bring problem domain expertise about the specific tasks that CASE might address.

- They should be an important source of how CASE choices may impact their immediate problems.

Organizational

- The organization is looking for a potential tool suite or SEE which could be available from organizational level and from which projects can select.
- The organization brings standards and management abilities to defining the problem space.
- The organization seeks to help define the software development process first, before potentially expensive CASE tools become a de-facto corporate standard.

Enterprise

- The enterprise should examine how do the CASE choices relate to long-term strategy.

8.4.2 Acquisition Strategy

Definition Acquisition strategy relates to elements of an overall plan aimed at buying CASE tools and corresponding supporting elements in a logical and coherent fashion.

Sub-Issues

- Examination of what level of cost benefits analysis need to be performed.
- Examination of the issue of building in-house versus buying from a commercial source versus buying and then tailoring the CASE tool.
- Examination of what existing tool can be re-used or should be redeployed as a result of obtaining new CASE tools.
- Examination of the effect of introducing CASE and its impact on a project's schedule versus time to execute an overall organizational-enterprise CASE adoption strategy.
- Examination of the need for and depth of methodology training required to efficiently use a CASE tool.
- Examination of the tool training required to understand and operate efficiently the mechanics of a CASE tool (e.g., user interface, database structures).
- Examination of acquisition time required.
- Examination of getting tool in-house for evaluation and plan for evaluation.
- Examination of negative productivity impact with untried tools.
- Examination of long- and short-term funding issues.

Scope

Project

- For projects, it should be advantageous to use tools or SEE supported by organization and or enterprise.
- Project should be cautioned when they wish to purchase project specific tools.

Organizational

- Organizations should aim to provide a tool suite and or SEE for use across many projects and problem domains.

Enterprise

- Enterprise should focus on broad acquisition strategies.
- Enterprise should negotiate corporate purchase agreements or GSA schedules.
- Enterprise should streamline purchasing processes.

8.4.3 Adoption Strategy

Adoption strategy is an important issue to be considered in an overall CASE Selection Strategy discussion, but discussion here was limited. This was due to a previous workshop which addressed these specific issues in-depth. Readers are encouraged to examine the results from this workshop (refer *Proceedings of CASE Adoption Workshop,* CMU/SEI-TR-91-14).

8.4.4 Readiness

Again, readiness is a very important topic, but discussion in this area was also limited due to a parallel workshop session dealing with the readiness issues. Please refer to CASE Readiness in Chapter 4 for the discussion and results of that session on readiness.

8.4.5 Process Control and Enforcement

Definition Process control and enforcement refers to techniques and practices which insure a software development process is controlled and adhered to.

Sub-Issues

- From a configuration management viewpoint, does the tool and its products lend themselves to CM, SEE CM and CM processes?
- Examination of process security issues like integrity, confidentiality and assurance of service.
- Existence of a process control policy and an architecture for supporting that policy.

Scope

Project

- Projects need to refine and implement organizational level CM and security policies.

Organizational

- Organizations need to determine how they want CM carried out.
- Organizations need to refine enterprise-level policies.
- Organizations need to develop security architectures for SEEs.

Enterprise

- Enterprise is responsible for high-level policy making.
- Enterprise policies are often guided by national requirements (e.g., level C2 security by 1992 for all federal work).

8.4.6 Software Engineering Environments

Definition A Software Engineering Environment (SEE) is a framework for tools and platforms to operate efficiently together, adding positive value to the task of the software development.

Sub-Issues

- Is the SEE being considered a fresh start or does it build upon an existing SEE?
- Does there exist an organizational-wide tool and platform standard?
- What degree of "openness" is desired in the SEE? (This influences the degree of cross vendor and platform compatibility required.)
- Is the SEE aimed at using commercially provided point-to-point tool integration techniques (possibly developed by coalitions of CASE vendors).
- What is the desired level of CASE data accessibility (e.g., import and export capabilities, granularity of data)?
- Will data elements in the SEE be fine or large-grained?
- What strategies and mechanisms will the SEE use to operate and control data and to integrate and control tools?

Scope

Project

- For small projects, it is advantageous to fit tool into existing SEE strategy.
- Project need to consider the impact of "fitting" tool into the SEE and what development or maintenance of tool/SEE interfaces.

Organizational

- Organizations should provide a SEE strategy that fits anticipated project needs.
- Organizations should consider advantages and limitations of available SEE frameworks.

Enterprise

- Enterprise should recognize that SEE standardization is most likely to be spotty and partial.
- Enterprises need to consider long term framework strategy.

8.4.7 Standards

Definition This refers to a range of applicable CASE related standards which may be defined at the local, Federal, and international levels.

Sub-Issues

- Awareness of the CASE standards sponsored by IEEE, ISO, and Military (MIL-STD).
- Awareness of corporate-developed or endorsed CASE standards.
- Awareness of customer-required CASE standards.

Scope

Project

- Projects need to balance enforcement of potentially incompatible CASE standards as levied by the customer and organization-enterprise standard. This assumes that the project is responsible for selecting and adhering to CASE related standards.

Organizational-Enterprise (depending upon the size of the overall corporate entity)

- Determining which CASE standards will be adhered to.
- Determining the benefits of standards versus the potential limiting in innovation brought about by some standards.
- Recognizing that not all standards are equal (e.g., many standards exist, but not all are actually supported by commercial CASE tools.)
- Responsible for tracking emerging and de-facto CASE standards.

8.4.8 Standard Practices

Definition This refers to commonly adhered to or formally defined corporate procedures that direct the organization's tool adoption process.

Sub-Issues

- Who is responsible for and does the tool selection?

- What is the organization's "history" of successes and failures in selecting new tools?
- What are the needs for document output generation from the tool?
- What is the existing selection process for tools and is it adequate?

Scope

Organizational

- Organizations should not re-invent selection process every time a new tool is acquired.

Enterprise

- Enterprise is responsible for the coordination of tool evaluations.
- At the enterprise level, a resource can be provided to provide information on tool selection process and to provide a repository of tool information (e.g., which tools have been examined and which tools are employed by units of the enterprise.)

8.5 Summary

This session on tool selection has focused upon a brief overview of the tool decision process. The primary emphasis of this session was to focus on strategies to discriminate between tools based upon the needs of different organization levels. The principle areas of discussion included strategies for problem space determination, tool acquisition, tool adoption, process control and enforcement, tool incorporation into a Software Engineering Environment, and tool adherence to a wide spectrum of standards and standard practices. Three different organizational levels were considered—project, organization, and enterprise. These levels formed the basis for focusing and narrowing the scope of tool selection issues considered in this session.

The strategies provided here are not intended to provide "all the answers" but to raise important and diverse sets of issues to be considered as organization sets about developing its own CASE selection strategy.

Acknowledgments

Many thanks to the session leaders, workshop participants and other key contributors to this workshop, and to reviewers of this paper, in particular, Sandy Brenner, Bill Curtis, Richard Martin, Joe Morin, Jock Rader, Dave Kitson, Anita Carleton, Suresh Konda, Jack Hilbing, Kurt Haverstock, John Antonucci, Marica Theoret, Jo Donatelli, Jane DeSimone, Andy Tsounos, Kurt Wallnau, Gibbie Hart, Grace Downey, Wendy Gissendanner, Helen Joyce, Wendy Rossi, Tracy Lewis, Attilio Marini, Frank Authenreith, Jeff Herman, Ed Morris, Sandy Brenner, Lorrie Orndoff, John Hollis, James Picone, Larry Crowe, Bob Kubiak, Randy Vandermolen, Rich Fox, Dave Vavrek, and to the technical writer/editor Bill McSteen.

References

[CASE 91]	CASE Consulting Group, Inc. "Guide to Products and Services" *CASE Outlook*. First edition, November 1991.
[Firth 87]	Firth, R.; Mosley, V.; Pethia, R.; Roberts Gold, L.; & Wood, W. *A Guide to the Classification and Assessment of Software Engineering Tools* (CMU/SEI-87-TR-10, DTIC: ADA213968). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University, 1987.
[FURPS 90]	Software Engineering Standards Subcommittee of the IEEE Computer Society. *Standard for a Software Quality Metrics and Methodology.* (April 1990): 73-75.
[Huff 91]	Huff, C.; Smith, D.; Stepien, K.; & Morris, E. Proceedings of the CASE Adoption Workshop (CMU/SEI-91-TR-14). Pittsburgh, Pa.: Software Engineering Institute, Carnegie Mellon University 1991.
[Humphrey 89]	Humphrey, W., *Managing the Software Process*. Reading, Mass: Addison-Wesley. 1989.
[Mosley 89]	Mosely, D., "Are We Ready for CASE." *American Programmer 2*, 5 (March 1989):21-26.
[Rubin 90]	Rubin, H., "How to become Software Engineering "Bigfoot." *American Programmer 3*, 1 (January 1990):22-32.
[Rubin 91]	Rubin, H. "CASE Champion's Toolkit—In Search of the True Cost of CASE." *CASE Outlook, 2* (1991).
[STSC 91]	Hanrahan, B.; van Buren, J.; Rieping, C.; Fujita-Yuhas, T.; Grotzky, J., Jones, G.; & Peterson, J. *Requirements Analysis & Design*. Hill Air Force Base, Ut.: Software Technology Support Center, 1991.

Appendix A CASE Market Overview

These following visuals are intended to give a quick overview of the current diverse CASE market. The data used to construct these charts and graphs was derived from the 1991 CASE OUTLOOK Guide to Products and Services [CASE 91]. This guide lists more than 800 CASE-related products.

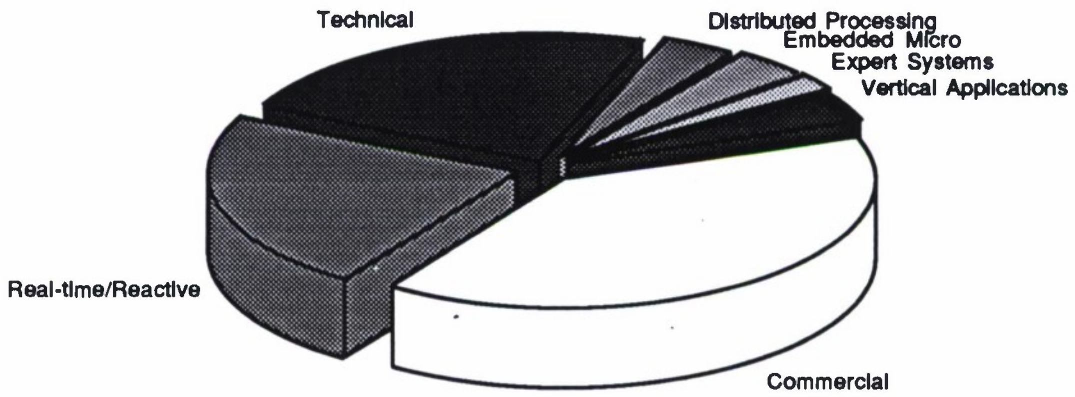


Figure A-1: CASE Market Overview

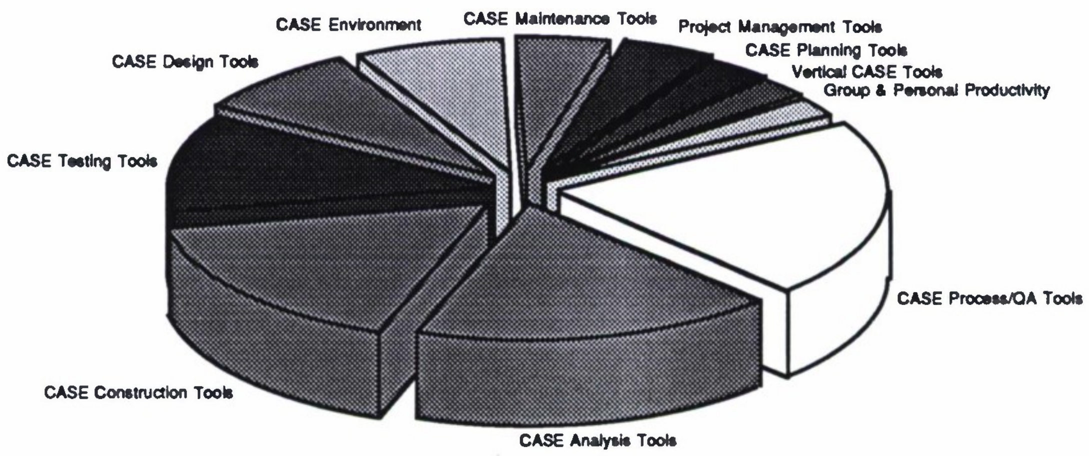


Figure A-2: CASE Market Categories

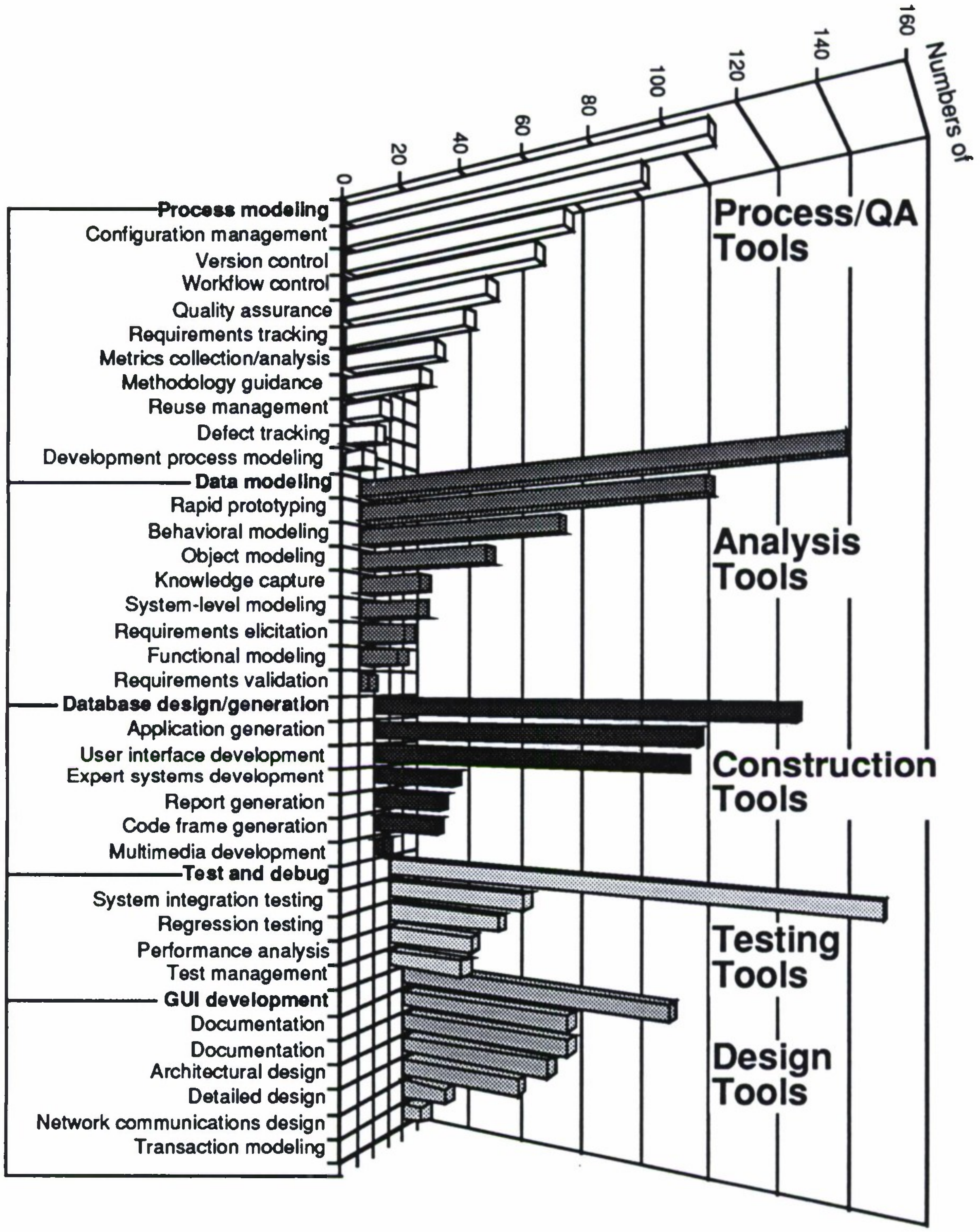


Figure A-3: CASE Market Category Detail Part 1

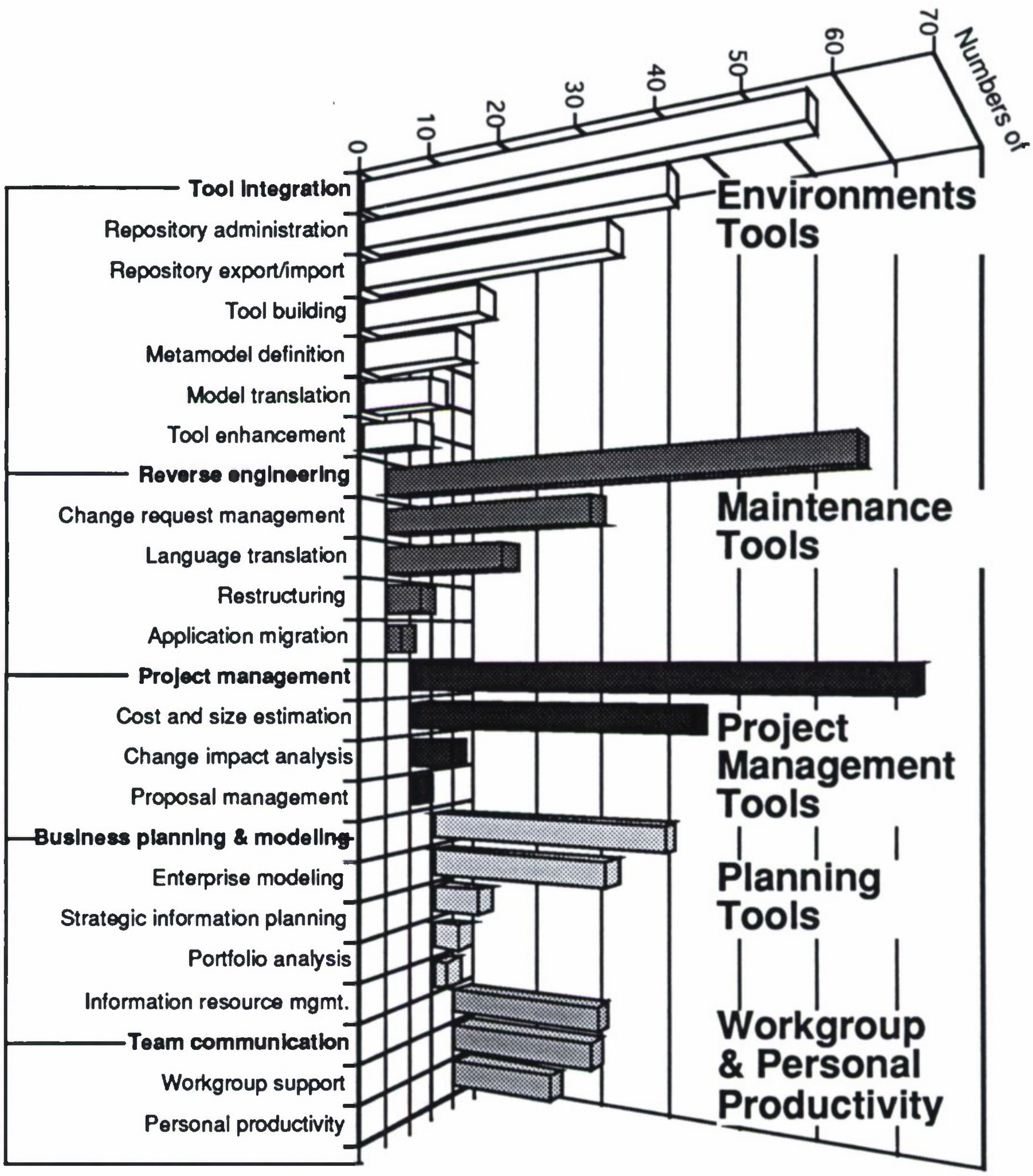


Figure A-4: CASE Market Category Detail Part 2

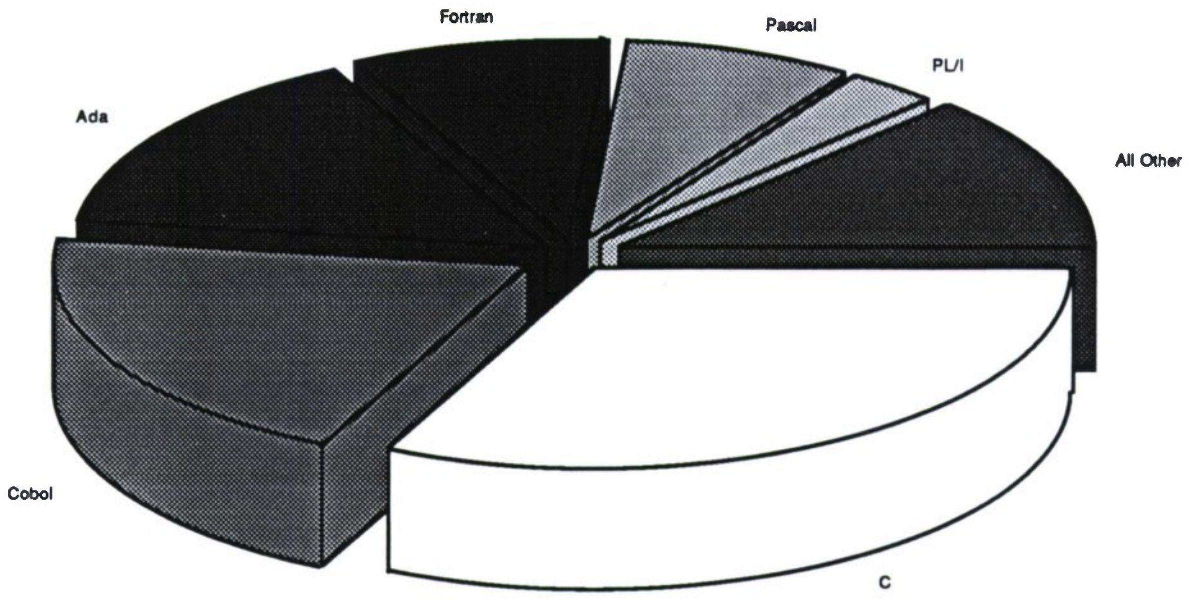


Figure A-5: Supported Languages

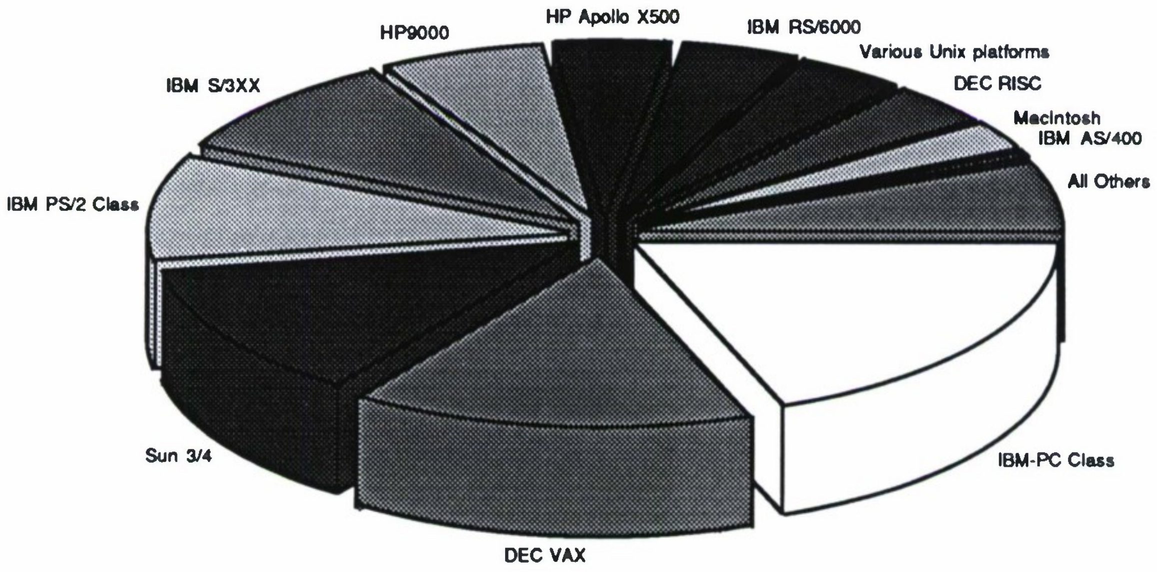


Figure A-6: Supported CASE Hardware Platforms

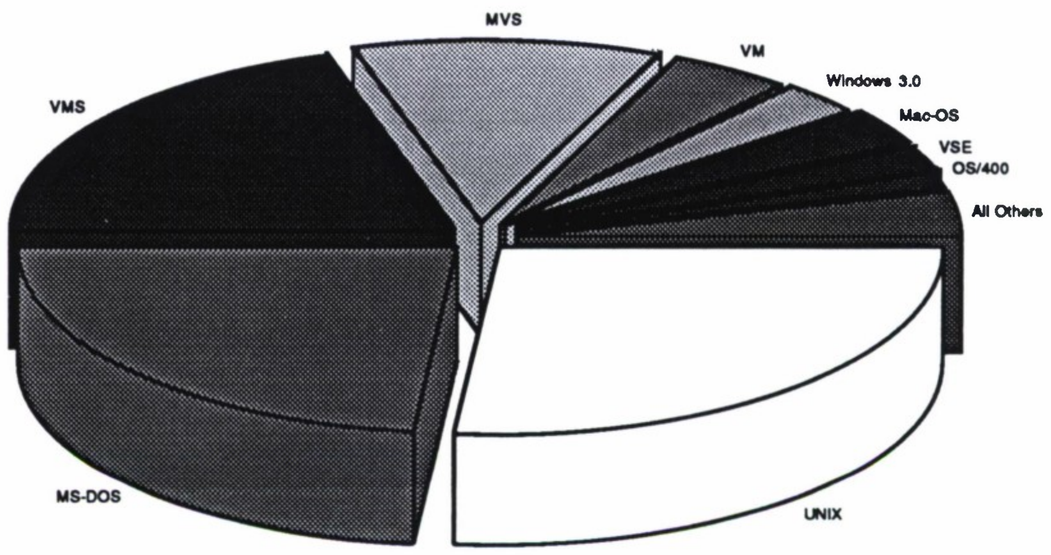


Figure A-7: Supported CASE Operating Systems

Appendix B Registration List

Frank Acello
Manager, Corporate CASE Initiative
Hughes Aircraft Company
Space & Communications Group
Building S64; MS-C409
P.O. Box 92919
Los Angeles, CA 90009
(213) 414-6229
FAX: (213) 414-6699

Lucy K. Aharonian
Senior Consultant
ANALYTICA
P.O. Box 403
Weston, MA 02193
(617) 891-1886

Bruce Allgood
Electronics Engineer
Software Technology Support Center
OO-ALC/TISAC
Hill AFB, UT 84056
(801) 777-7703
FAX: (801) 777-8069

Jerry Baum
Senior Scientist, CASE Project
Hughes Aircraft Company
Space & Communications Group
(SC/S64/C409)
P.O. Box 92919
Los Angeles, CA 90009
(213) 414-6241
jbaum@luna.dpl.scg.hac.com
FAX: (213) 414-6699

Kevin J. Berk
SEE Team Leader
Software Technology Support Center
OO-ALC/TISAC
Hill AFB, UT 84056
(801) 777-7703
berk@oodis01.af.mil
FAX: (801) 777-8069

Jack Bond
Software Engineering Staff
National Security Agency
ATTN: T303
9800 Savage Road
Fort George Meade, MD 20755-6000
(301) 688-7691

Odean Bowler
Software Engineer
Software Technology Support Center
CASE Tools/Environments
OO-ALC/TISAC Bldg. 100
Hill AFB, UT 84056
(801) 777-8045
FAX: (801) 777-8069

Sandy Brenner
Secretary III, Case & SAE Projects
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-3444
seb@sei.cmu.edu
FAX: (412) 268-5758

Anita D. Carleton
Project Manager
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7718
adc@sei.cmu.edu
FAX: (412) 268-5758

Charles B. Cavanaugh
Senior Marketing Representative
IBM Corporation
Application Solutions Division
1503 LBJ Freeway
Dallas, TX 75234
(214) 406-7632
FAX: (214) 406-7483

Batia Dane
Senior Member of Technical Staff
GTE
Govemment Systems Corp.
77 A Street
Needham Heights, MA 02194-2892
(617) 455-5366
FAX: (617) 435-5365

Tina M. DeAngelis
Graduate Student, Computer Systems
United States Air Force
696G
Treasury Drive
Kettering, OH 45429
(513) 255-8989
tdeangel@galaxy.afit.af.mil

Jane Walter DeSimone
Program Administrator
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7580
jwd@sei.cmu.edu
FAX: (412) 268-5758

Anna Deeds
NSS/SECR Acquisitions Manager
Naval Sea Systems Command
PMS 412
2351 Jefferson Davis Highway
Room 11E28
Arlington, VA 22202
(703) 602-8204
FAX: (703) 602-2070

Grace F. Downey
Member of Technical Staff
Software Engineering Institute
TABS/CF
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7601
downey@sei.cmu.edu
FAX: (412) 268-5758

Walter DuBlanica
Consulting Engineer
ETSS
300 Harper Place
Suite 201, Building 2
Moorestown, NJ 08057
(609) 273-6666
FAX: (609) 727-9770

Greg Engledove
Computer Systems Analyst
Naval Sea Systems Command
PMS-412
2351 Jefferson Davis Highway
Room 11E28
Arlington, VA 22202
(703) 602-8204
FAX: (703) 602-2070

Susan M. Frankhuizen
Software Engineer
IBM Corporation
Federal Sector Division
FSI
9500 Goodwin Drive
Mail Stop 101/087
Manassas, VA 22110
(703) 367-2514
FAX: (703) 367-4039

Glenn Harmon
Air Staff SW Manager
United States Air Force
HQ USAF/SCXS
Washington, DC 20330
(703) 614-7027
harmon@sc4.hq.af.mil
FAX: (703) 695-4022

Gibbie Lu Hart
Computing Facilities Manager
Software Engineering Institute
Products & Services
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7780
gibbie@sei.cmu.edu
FAX: (412) 268-5758

Donald F. Heitzmann
VP of Engineering
Cadre Technologies, Inc.
Teamwork Division
222 Richmond Street
Suite 301
Providence, RI 02903
(401) 351-5950

Jeffrey Herman
Resident Affiliate
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-8738
jrh@sei.cmu.edu
FAX: (412) 268-5758

Jack Hilbing
Director, Technical & Business Services
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7626
fjh@sei.cmu.edu
FAX: (412) 268-5758

Clifford C. Huff
Member of Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7605
cch@sei.cmu.edu
FAX: (412) 268-5758

Sok Kim
Software Engineer
US Army CECOM
Software Engineering Technology Branch
Building 1209
Fort Monmouth, NJ 07703
(908) 532-2146
kims@ajpo.cmu.sei.edu
FAX: (908) 532-4129

David Kitson
Senior Researcher
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7782
dhk@sei.cmu.edu
FAX: (412) 268-5758

Suresh Konda
Research Associate
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-8783
slk@sei.cmu.edu
FAX: (412) 268-5758

Randall W. Lichota
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-6834
rwl@sei.cmu.edu
FAX: (412) 268-5758

Dick Martin
Member of the Technical Staff
Software Engineering Institute
Program Development
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-7617
martin@sei.cmu.edu
FAX: (412) 268-5758

Raymond Menell
Computer Scientist
US Army CECOM
AMSEL-RD-SE-AST-SE
Fort Monmouth, NJ 07703
(201) 542-4170

Paul Meserole
Computer Scientist
Naval Air Development Center
Code 7031
Warminster, PA 18974-5000
(215) 441-1261
meserole@nadc.nadc.navy.mil
FAX: (215) 441-3225

Joe Morin
Visiting Scientist
Software Engineering Institute
Products and Services
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-8594
jfm@sei.cmu.edu
FAX: (412) 268-5758

Ed Morris
Member of the Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-5754
ejm@sei.cmu.edu
FAX: (412) 268-5758

Kimberly Oakes
Computer Scientist
Department of Defense
T34
9800 Savage Road
Ft. Meade, MD 20755-6000
(301) 688-7072

Gary Petersen
Technical Program Manager
Software Technology Support Center
OO-ALC/TISE
Hill AFB, UT 84056
(801) 777-7703
petersen@oodis01.af.mil
FAX: (801) 777-8069

Jock A. Rader
Sr. Lab Scientist
Hughes Aircraft Company
Radar Systems Group
Box 92426 (R8/5100)
Los Angeles, CA 90009
(213) 607-3488
jock@sdfvu9.dnet.hac.com
FAX: (213) 334-2693

Kenneth E. Rowe
National Security Agency
Attn: S9
9800 Savage Rd.
Fort Meade, MD 20755-6000
(410) 684-7374
rowe@dockmaster.ncsc.mil

Dennis B. Smith
MTS/Project Leader
Software Engineering Institute
Technology Division
Carnegie Mellon University
Room 5408
Pittsburgh, PA 15213-3890
(412) 268-6850
dbs@sei.cmu.edu
FAX: (412) 268-5758

Jay Stanley
Resident Affiliate
US Army CECOM
CSE-AMSEL-RD-SE-CCS
Ft. Monmouth, NJ 07703
(412) 268-5780
jcs@sei.cmu.edu
FAX: (412) 268-5758

Betty Topp
Graduate Student, Computer Systems
Air Force Institute of Technology
AFIT/ENA
5495 Gander Road South
Dayton, OH 45424
(513) 255-8989
btopp@galaxy.afit.af.mil

Andrew Tsounos
Member of Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-6304
agt@sei.cmu.edu
FAX: (412) 268-5758

Kurt C. Wallnau
Member of Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-3660
kcw@sei.cmu.edu
FAX: (412) 268-5758

Neal Walters
Senior Engineer
IBM Corporation
Federal Sector Division
FSD 250/059
9500 Godwin Drive
Manassas, VA 22110
(703) 367-3577
FAX: (703) 367-5067

Susan Warshaw
Computer Scientist
Defense Information Systems Agency
Center Information Management
Code XE
S. Courthouse Road
Washington, DC 20305
(703) 285-5310
FAX: (703) 285-5435

Paul Zarrella
Member of Technical Staff
Software Engineering Institute
Technology Division
Carnegie Mellon University
Pittsburgh, PA 15213-3890
(412) 268-3156
pfz@sei.cmu.edu
FAX: (412) 268-5758

Appendix C Workshop Session Assignments

C.1 CASE Acquisition Policy

1. Session Leader

Mr. Richard Martin	Software Engineering Institute
Mr. Joseph Morin	Software Engineering Institute

2. Scribe

Ms. Jane DeSimone	Software Engineering Institute

3. Participant

Captain Kevin Berk	Software Technology Support Center
Mr. Jack Bond	National Security Agency
Mr. Greg Engledove	Department of the Navy
Mr. Jack Hilbing	Software Engineering Institute
Capt Jeff Herman	US Army CECOM/SEI RA
Mr. Jay Stanley	US Army CECOM/SEI RA
Mr. Neal Walters	IBM

C.2 What CASE Tools Actually Do/What They Don't Do

1. Session Leader

Mr. J. A. Rader	Hughes Aircraft Company

2. Scribe[p

Mr. Andy Tsounos	Software Engineering Institute

3. Participant

Mr. Odean Bowler	Software Technology Support Center
Ms. Susan M. Frankhuizen	IBM
Mr. Donald Heitzmann	Cadre Technologies, Inc.
Mr. Randall Lichota	Hughes Aircraft Company
Mr. Ray Menell	US Army CECOM Center for Software
Ms. Kim Stepien Oakes	National Security Agency

C.3 CASE and Metrics

1. Session Leader

Mr. Ed Morris	Software Engineering Institute

2. Scribe

Mr. Kurt Wallnau	Software Engineering Institute

3. Participant

Ms. Anita Carleton	Software Engineering Institute
Ms. Anna Deeds	Department of the Navy
Mr. Walt DuBlanica	ETSS
Major Glenn Harmon	HQ USAF/SCXS
Mr. Suresh Konda	Software Engineering Institute
Mr. Paul Meserole	Naval Air Development Center

C.4 CASE Readiness

1. Session Leader

Mr. Dave Kitson	Software Engineering Institute

2. Scribe

Ms. Gibbie Hart	Software Engineering Institute

3. Participant

Mr. Frank Acello	Hughes Aircraft Company
Ms. Lucy K. Aharonian	Analytica
Mr. H. Bruce Allgood	Software Technology Support Center
Mr. Chuck Cavanaugh	IBM
Mr. Raymond Yeh	International Software Systems, Inc.
Mr. Paul Zarrella	Software Engineering Institute

C.5 CASE Tool Selection

1. Session Leader

Mr. Clifford Huff	Software Engineering Institute
Mr. Dennis Smith	Software Engineering Institute

2. Scribe

Ms. Grace Downey	Software Engineering Institute

3. Participant

Mr. Jerry Baum	Hughes Aircraft Company
Ms. Batia Dane	GTE Government Systems Corporation
Captain Tina M. DeAngelis	USAF
Mr. Sok Kim	US Army HQ CECOM, CSE
Mr. Gary Petersen	Software Technology Support Center
Mr. Ken Rowe	National Security Agency
Capt Betty Topp	Air Force Institute of Technology
Ms. Susan Warshaw	Defense Communications Agency

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS None
2a. SECURITY CLASSIFICATION AUTHORITY N/A 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE N/A	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release Distribution Unlimited
4. PERFORMING ORGANIZATION REPORT NUMBER(S) CMU/SEI-92-TR-6	5. MONITORING ORGANIZATION REPORT NUMBER(S) ESC-TR-92-006
6a. NAME OF PERFORMING ORGANIZATION Software Engineering Institute 6b. OFFICE SYMBOL (if applicable) SEI	7a. NAME OF MONITORING ORGANIZATION SEI Joint Program Office
6c. ADDRESS (city, state, and zip code) Carnegie Mellon University Pittsburgh PA 15213	7b. ADDRESS (city, state, and zip code) ESC/AVS Hanscom Air Force Base, MA 01731
8a. NAME OFFUNDING/SPONSORING ORGANIZATION SEI Joint Program Office 8b. OFFICE SYMBOL (if applicable) ESC/AVS	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F1962890C0003
8c. ADDRESS (city, state, and zip code)) Carnegie Mellon University Pittsburgh PA 15213	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO 63756E PROJECT NO. N/A TASK NO N/A WORK UNIT NO. N/A

11. TITLE (Include Security Classification)
Proceedings of the CASE Management Workshop

12. PERSONAL AUTHOR(S)
Cliff Huff Dennis Smith Ed Morris Paul Zarrella

13a. TYPE OF REPORT Final	13b. TIME COVERED FROM TO	14. DATE OF REPORT (year, month, day) September 1992	15. PAGE COUNT 79

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB. GR.

18. SUBJECT TERMS (continue on reverse of necessary and identify by block number)
CASE, Computer Aided Software Engineering, CASE Management,
CASE Forecast, Metrics, Readiness, Selection, Acquisition

19. ABSTRACT (continue on reverse if necessary and identify by block number)

The Software Engineering Institute (SEI) Computer-Aided Software Engineering (CASE) Technology Project sponsored a workshop to address a number of key CASE management issues. The workshop was held at the SEI in Pittsburgh, Pennsylvania on June 19-20, 1991. At the workshop, a representative group of SEI affiliates from industry, government, and academia discussed among themselves such management topics as CASE acquisition policy, what CASE tools can and cannot do, CASE and metrics, and CASE tool selection. The results of these discussions are summarized in this report.

(please turn over)

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ■ SAME AS RPT ☐ DTIC USERS ■	21. ABSTRACT SECURITY CLASSIFICATION Unclassified, Unlimited Distribution
22a. NAME OF RESPONSIBLE INDIVIDUAL Tom Miller, Lt Col, USAF	22b. TELEPHONE NUMBER (include area code) (412) 268-7631 22c. OFFICE SYMBOL ESC/AVS (SEI)

DD FORM 1473, 83 APR EDITION of 1 JAN 73 IS OBSOLETE

UNLIMITED, UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS